Android Application -- BuzzBee Design Document

Team:

GoldenBros

TeamMember:

Name:	**Andrew_ID:**
Zijun Wang	zijunwan
Shuai Wang	shuaiwa1
Ge Jin	gjin

Date:

Aug.3rd, 2015

Preview

3.Flowcharts

4.PageFlow Diagram

5.Class Diagram

6.Tiers Interaction Diagram

7.Designing Presentation Tier

8.Designing Content Provider(For Storage)

9.Designing Application Tier(For business logic)

10.Designing Integration Tier

11.Exception Handling

1.Introduction

1.1 Idea to create BuzzBee

Human beings is the creature that lives in social. There are always links between friends, families, workmates and even strangers. Aggregating a group of people for a certain purpose encourages them to make new friends, solidify relationship, and more important, run out of their own cycles. Nowadays, however, more and more people pay more attention to their own stuff. Especially with the booming of the Internet, people actually prefer staring at the screen and addicting to virtual world, which is apparently not good. Furthermore, It will lead to serious problems such as

less trust between people, society disharmony and unhealthy lifestyle. Facing such severe problem, we teammates are planning to develop an mobile-app, which has an interesting name “BuzzBee”, to help people return to essence of life, live in a positive lifestyle, and spend spare time reasonably and significantly.

1.2 Name of BuzzBee

As far as we know, bees are gregarious animal and they never do stuff alone. Besides, human beings are also gregarious creature. Based on that fact, we associate people with bees to highlight the theme of our application: aggregation.

This lovely and pellucid name shrink the distance between machine and human beings. When user see it, they know what does it means and eager to try because of its lovely UI. This is not just an application, but it represents positive attitude towards life. Although BuzzBee is just a software built in hardware, user can figure out the emotion inside it and find the essence of true life.

1.3 What’s it about?

One can regard BuzzBee as a platform that help people find interested group, such as basketball group, volunteer group and computer group. People with same habit can do whatever they like together. In this case, people will not spend most of their spare time in virtual world, they can reunderstand the essence of life, and lives more happier.Each user can issue and join an activity. We’ll also have a mechanism to allow activity holders to limit how many and what kinds of user could participate in the activity. One can also use BuzzBee as a medium to release important notices. For instance, CMU organization can issue a post to call on people to take part in one seminar.

1.4 Who’re users?

Those people who have no idea what to do for spare time, who want to spend more time on true life instead of virtual world, who want to know

more about others, who want to widen their social network, who want to perform activities with a group of people are all potential users.

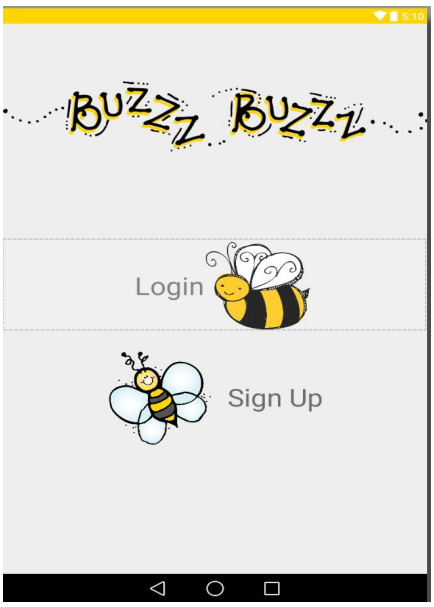
2.Designing Screens

1> nexus 7 2012: 1920 x 1200 resolution and 323ppi pixel density, 7.0 inches screen size(Large screen, Extra High Density)

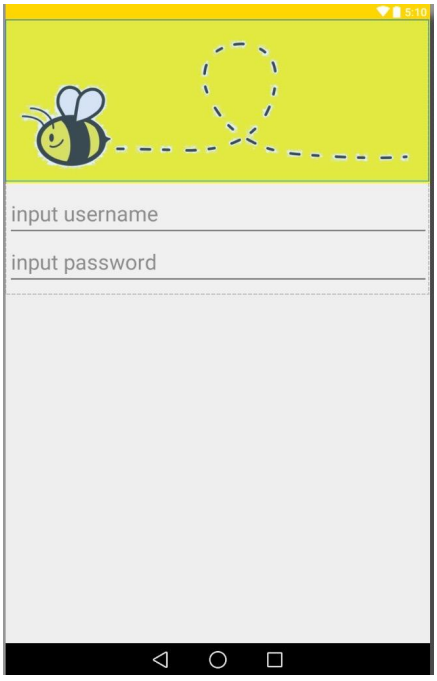
2> nexus S: 800 x 480 px(0.37 megapixels) and 233 ppi, 4.0 inches screen size(Normal screen, High Density)

Following is UI example for large screen. Because of all ui is same for large screen and normal screen, we only present large screen.

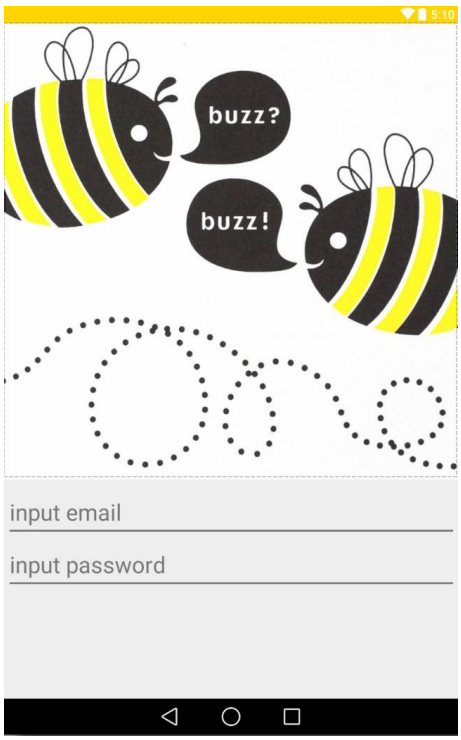
Login Page -- When you start BuzzBee app, this is the first page you'll see.



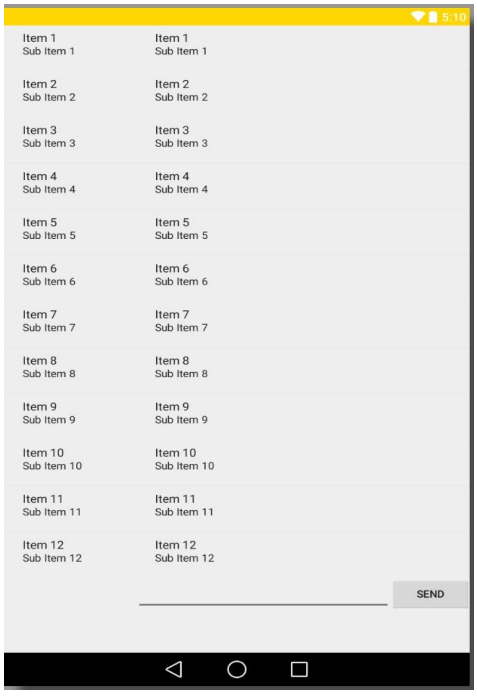
Login Dialog

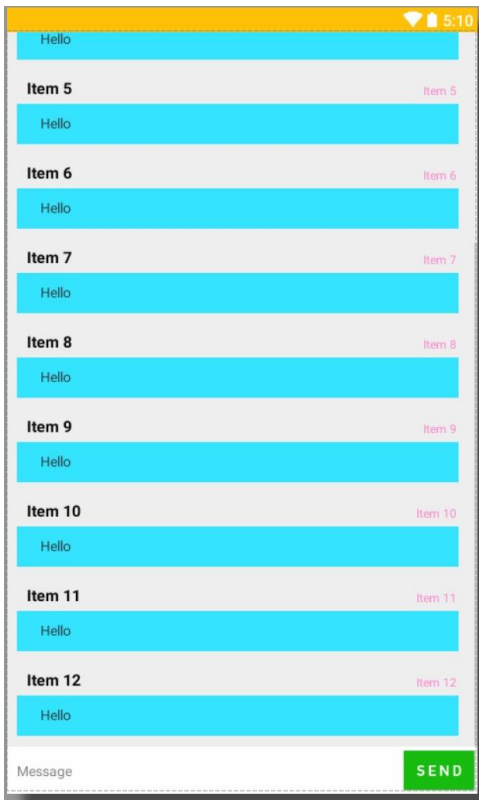


Signup Dialog

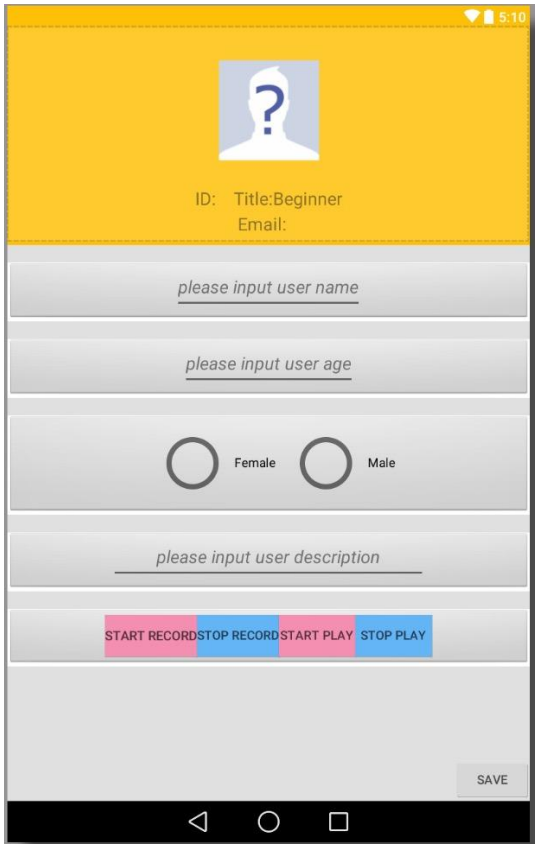


Chatroom Page -- friends list is shown on left side, right side is char message.

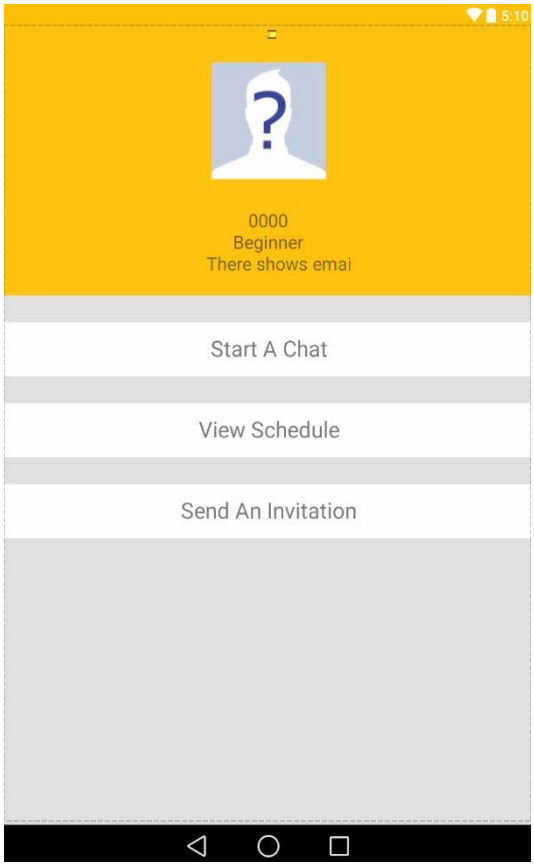




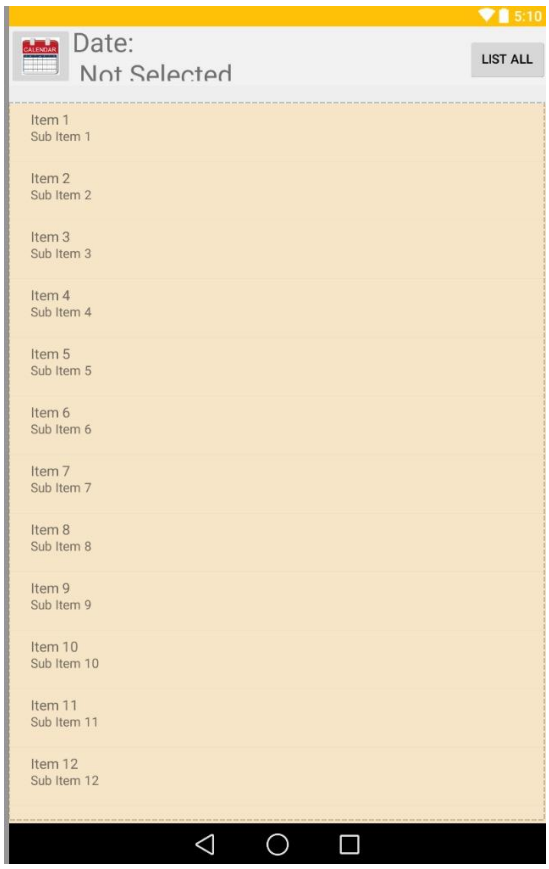
Personal Information Page.(this is just an example)



Friend Information Page.(this is just an example) User can chat with friends or invite friends to an activity.



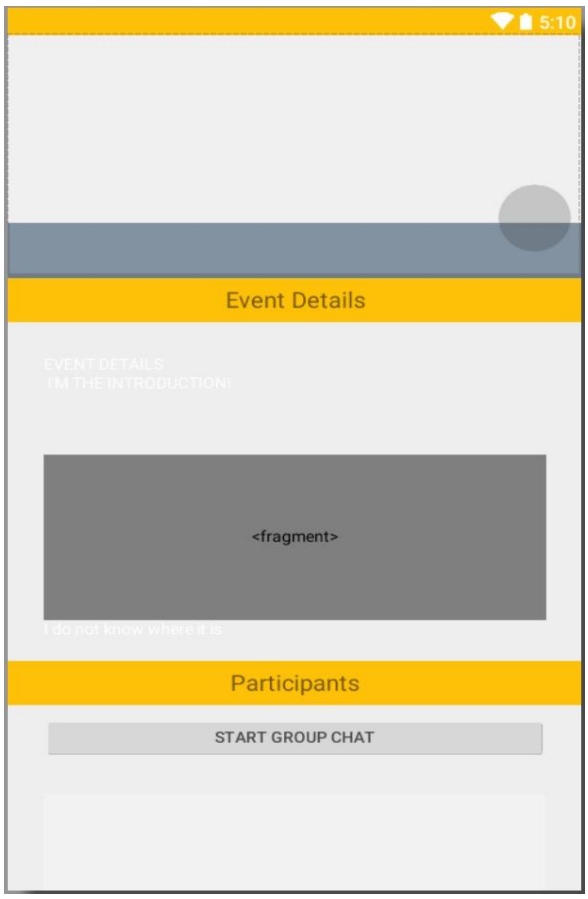
Schedule Page -- User can check all activities he/she has joined or issued.

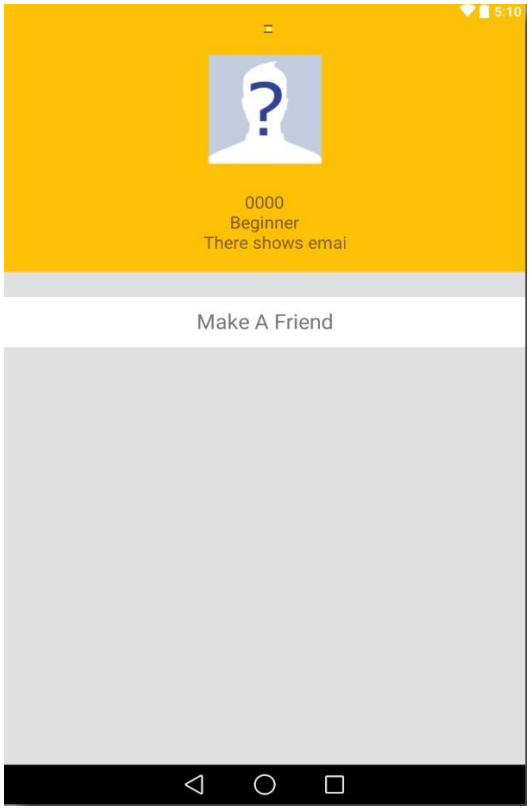


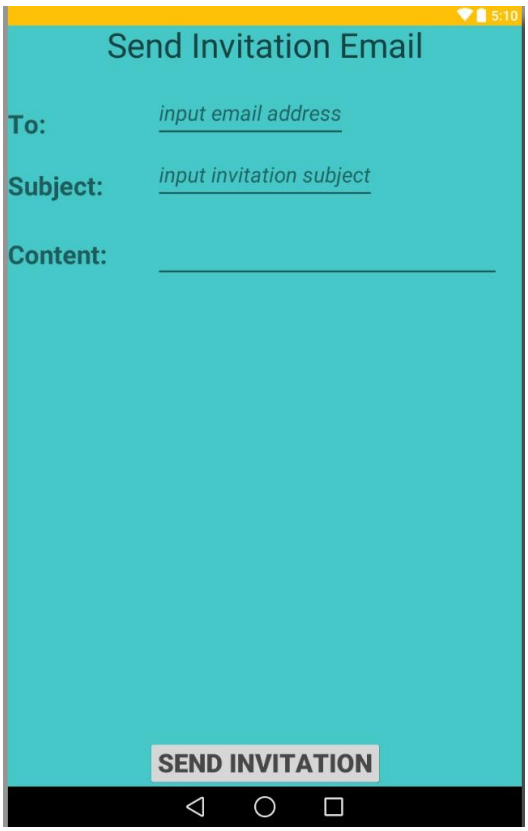
Event Page -- User can choose to issue or join an activity.

Find an Event -- User can input filter conditions to view certain events he/she prefers.

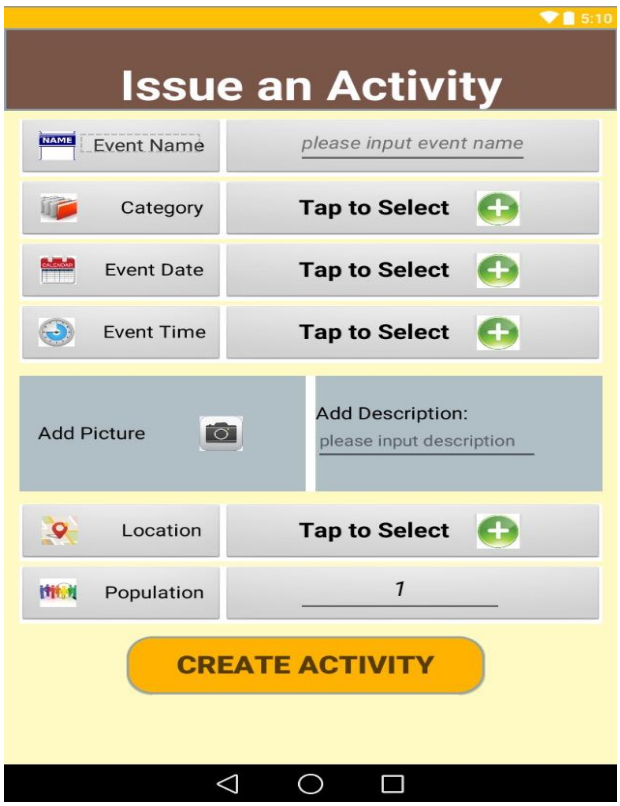
User can choose one event from events list, then this page will show. User can see detailed information here, and decide join or nor. If an user joined this event, he/she can enter chatroom directly from here.


5:10
Event Details
EVENT DETAILS
I'M THE INTRODUCTION!
<fragment>
I do not know where it is
Participants
START GROUP CHAT

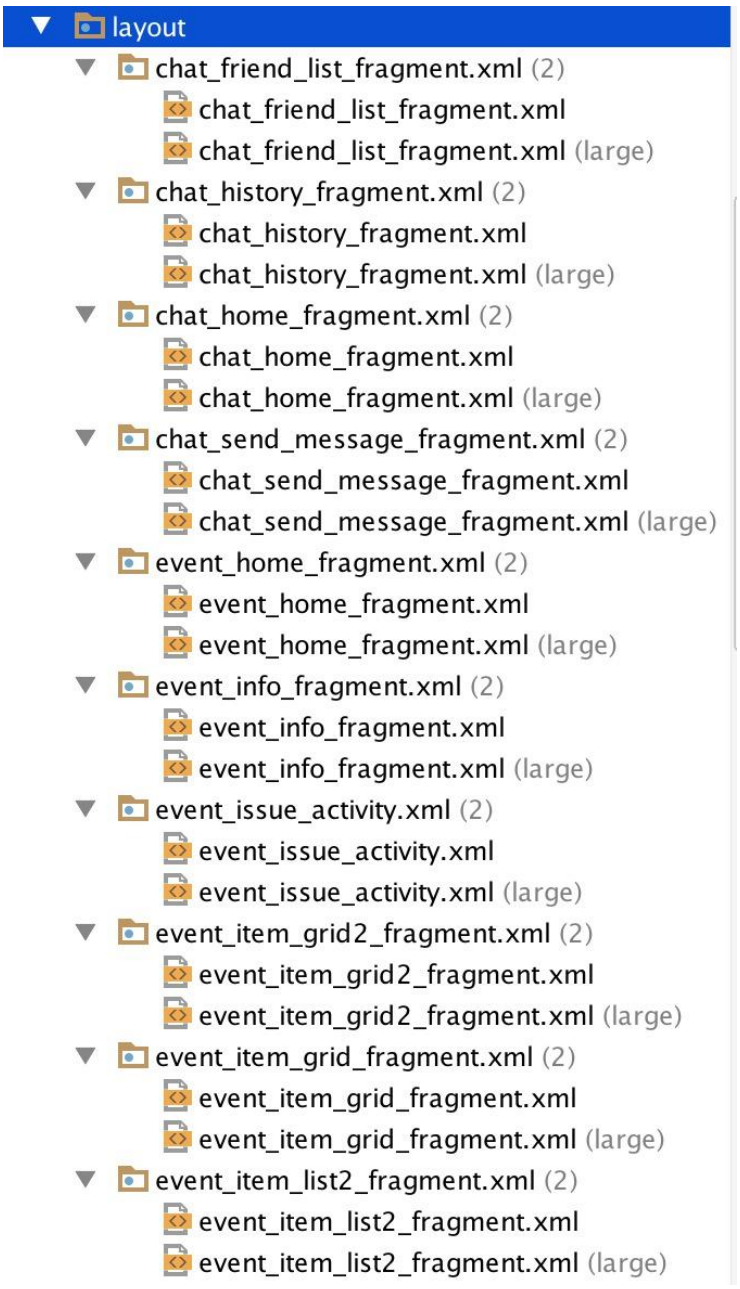

5:10
0000
Beginner
There shows emai
Make A Friend

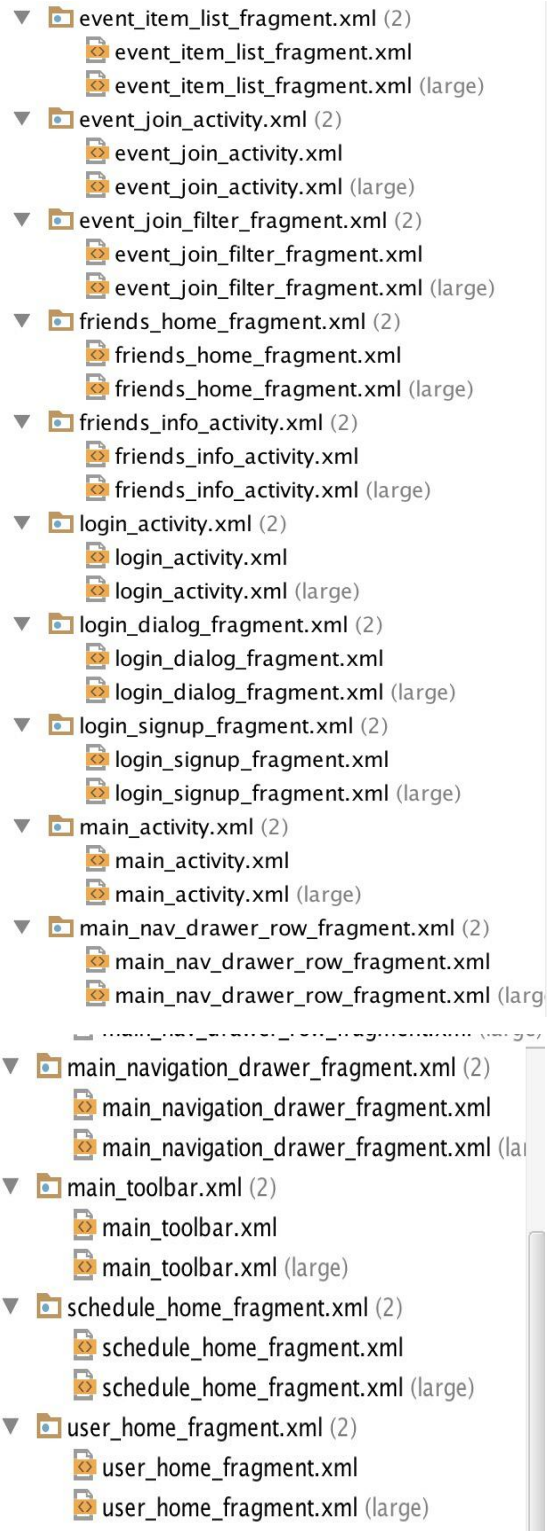


In this page, user can issue an activity. User must input detailed information.



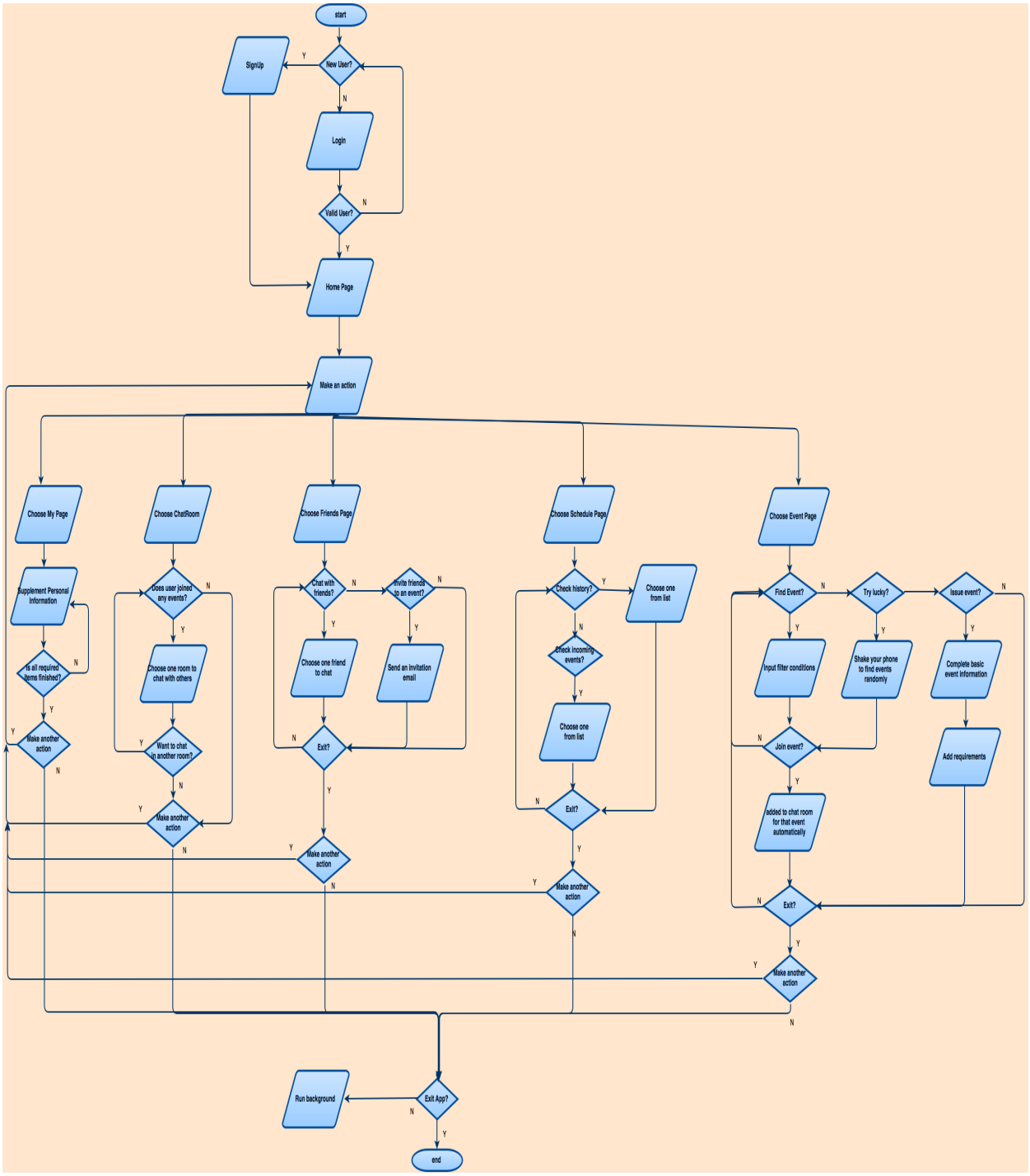
There are layout folder to indicate we can support two different size screens.


layout
chat_friend_list_fragment.xml (2)
chat_friend_list_fragment.xml
chat_friend_list_fragment.xml (large)
chat_history_fragment.xml (2)
chat_history_fragment.xml
chat_history_fragment.xml (large)
chat_home_fragment.xml (2)
chat_home_fragment.xml
chat_home_fragment.xml (large)
chat_send_message_fragment.xml (2)
chat_send_message_fragment.xml
chat_send_message_fragment.xml (large)
event_home_fragment.xml (2)
event_home_fragment.xml
event_home_fragment.xml (large)
event_info_fragment.xml (2)
event_info_fragment.xml
event_info_fragment.xml (large)
event_issue_activity.xml (2)
event_issue_activity.xml
event_issue_activity.xml (large)
event_item_grid2_fragment.xml (2)
event_item_grid2_fragment.xml
event_item_grid2_fragment.xml (large)
event_item_grid_fragment.xml (2)
event_item_grid_fragment.xml
event_item_grid_fragment.xml (large)
event_item_list2_fragment.xml (2)
event_item_list2_fragment.xml
event_item_list2_fragment.xml (large)



3.Flowcharts

This part helps you have a whole idea of our application from start to end.

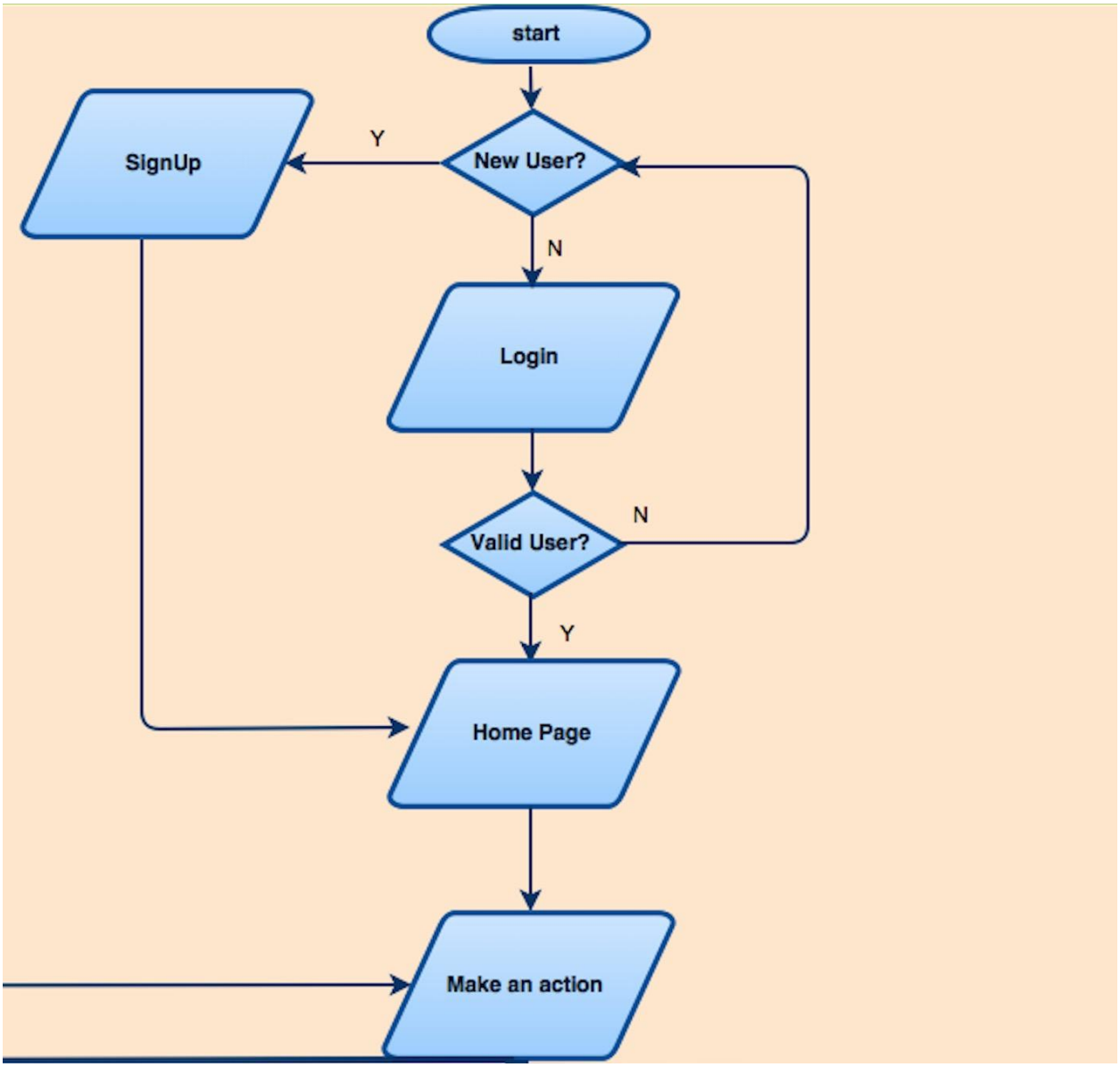


The following pictures are part of the whole flowcharts:

3.1 start part

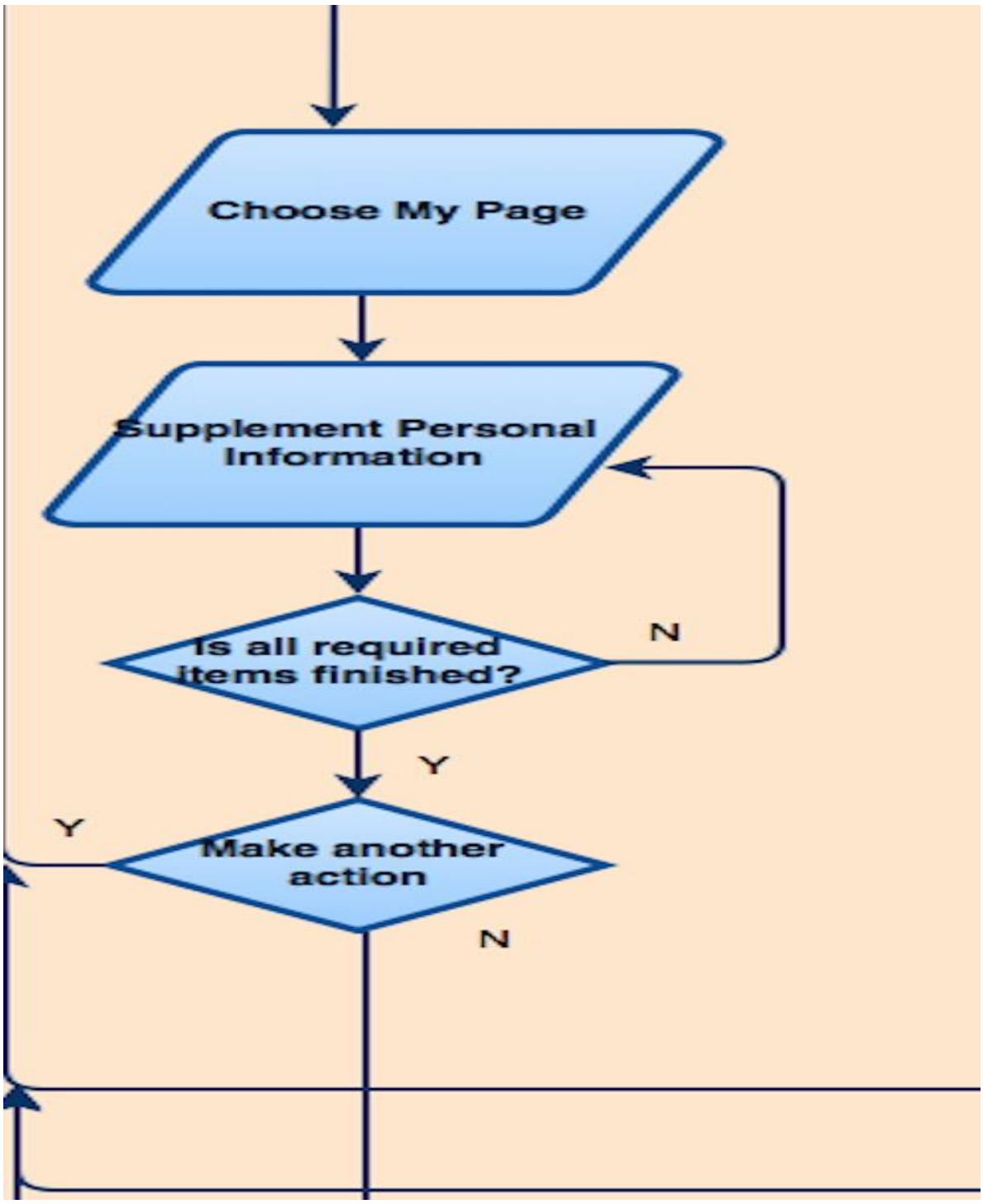
This pictures shows the start part of BuzzBee. User can login or signup to enter the home page, where he/she can take 5 types of different actions: check user information, enter chatroom, see friends, check schedule and

take events.



3.2 My Page

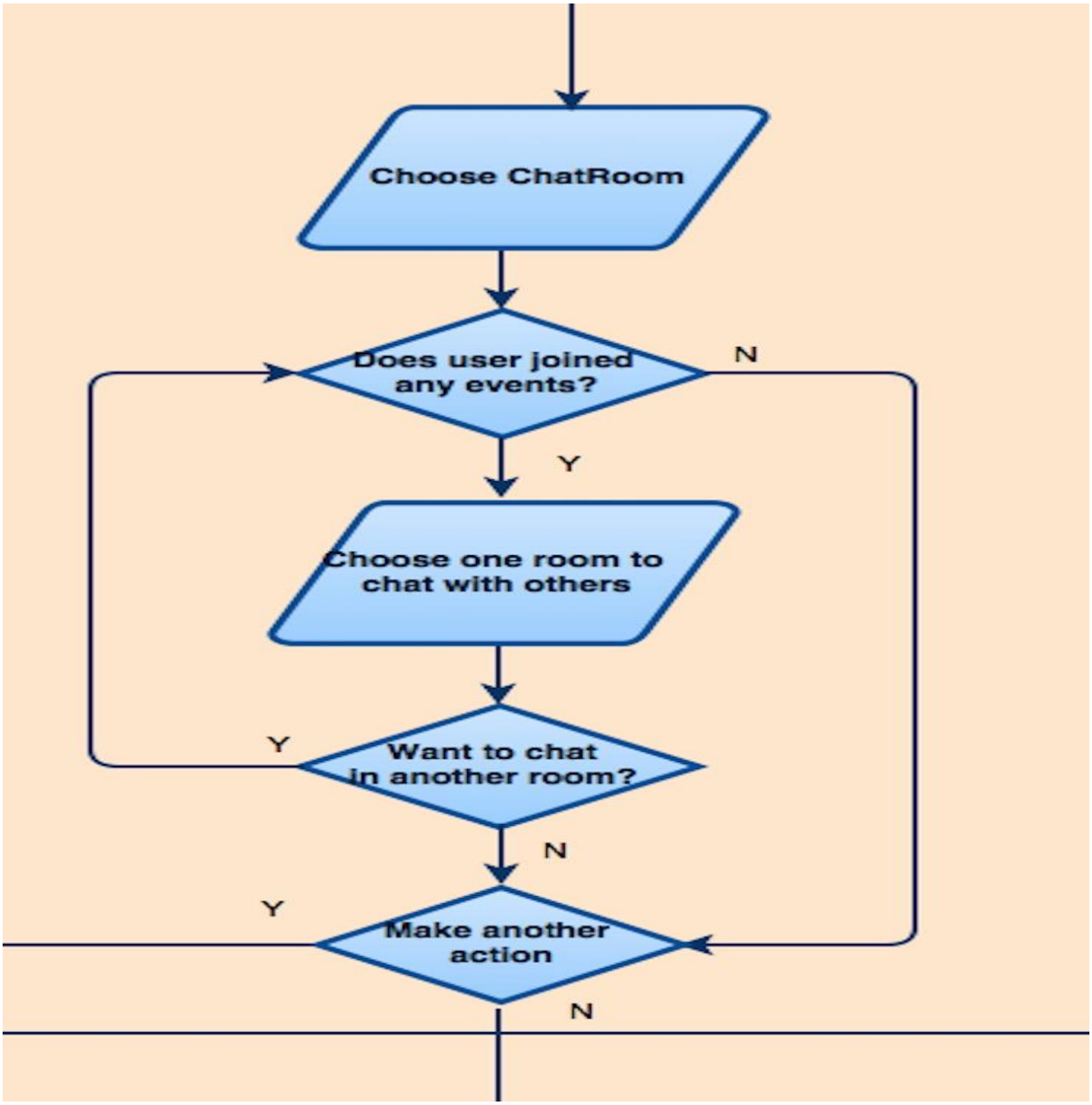
User can choose to enter user information page. When user is in this page, he/she can check, supplement and edit personal information. After that, user can choose to make another action.



3.3 Chat Room

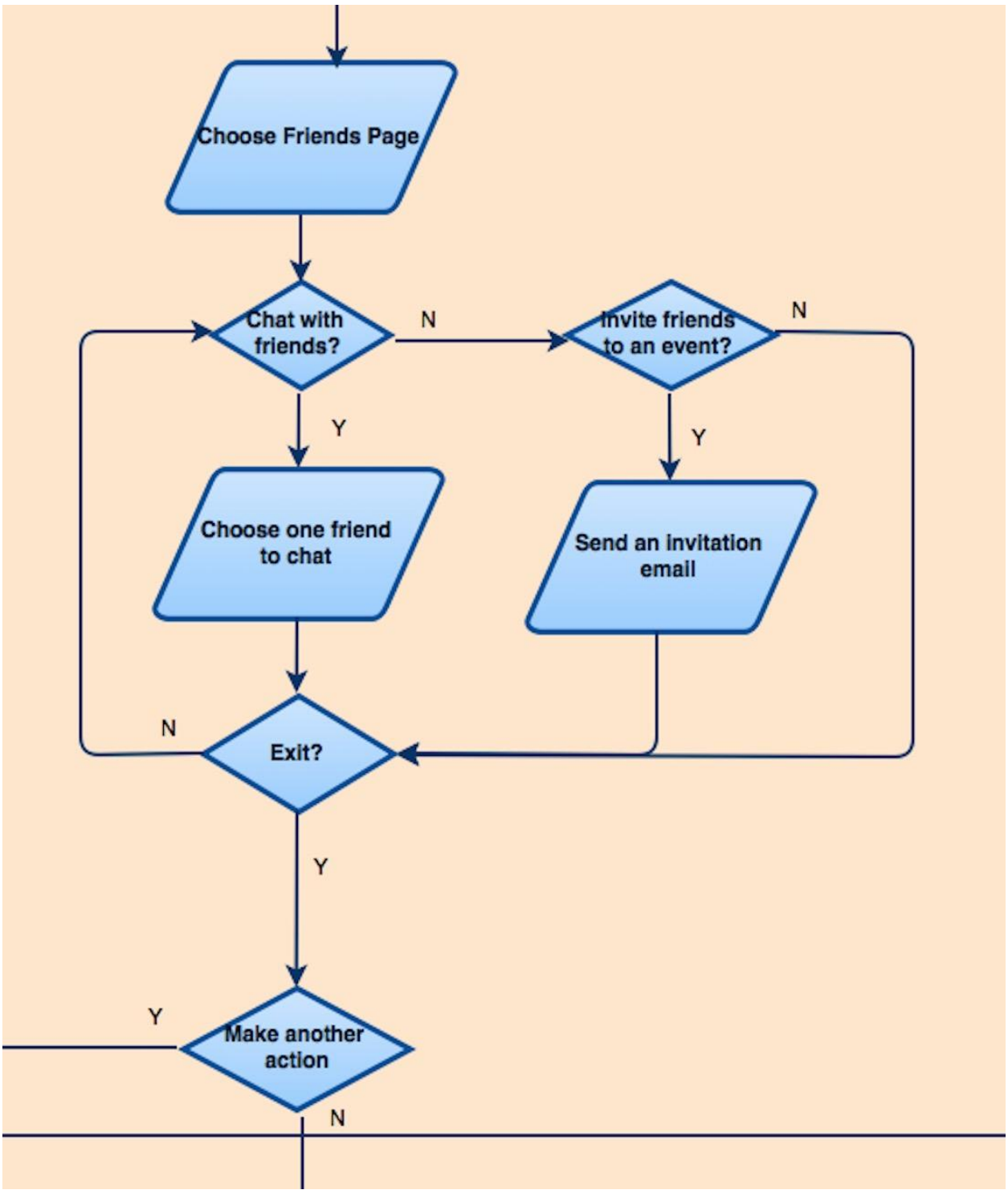
User can choose to enter chatroom. If the user has joined some activities, he/she can see the corresponding chatroom list in Chat-Room Page. User can click to enter one of the chatrooms to chat with all people who joined

the same activity. Also, he/she can choose to chat in another chatroom or make another action.



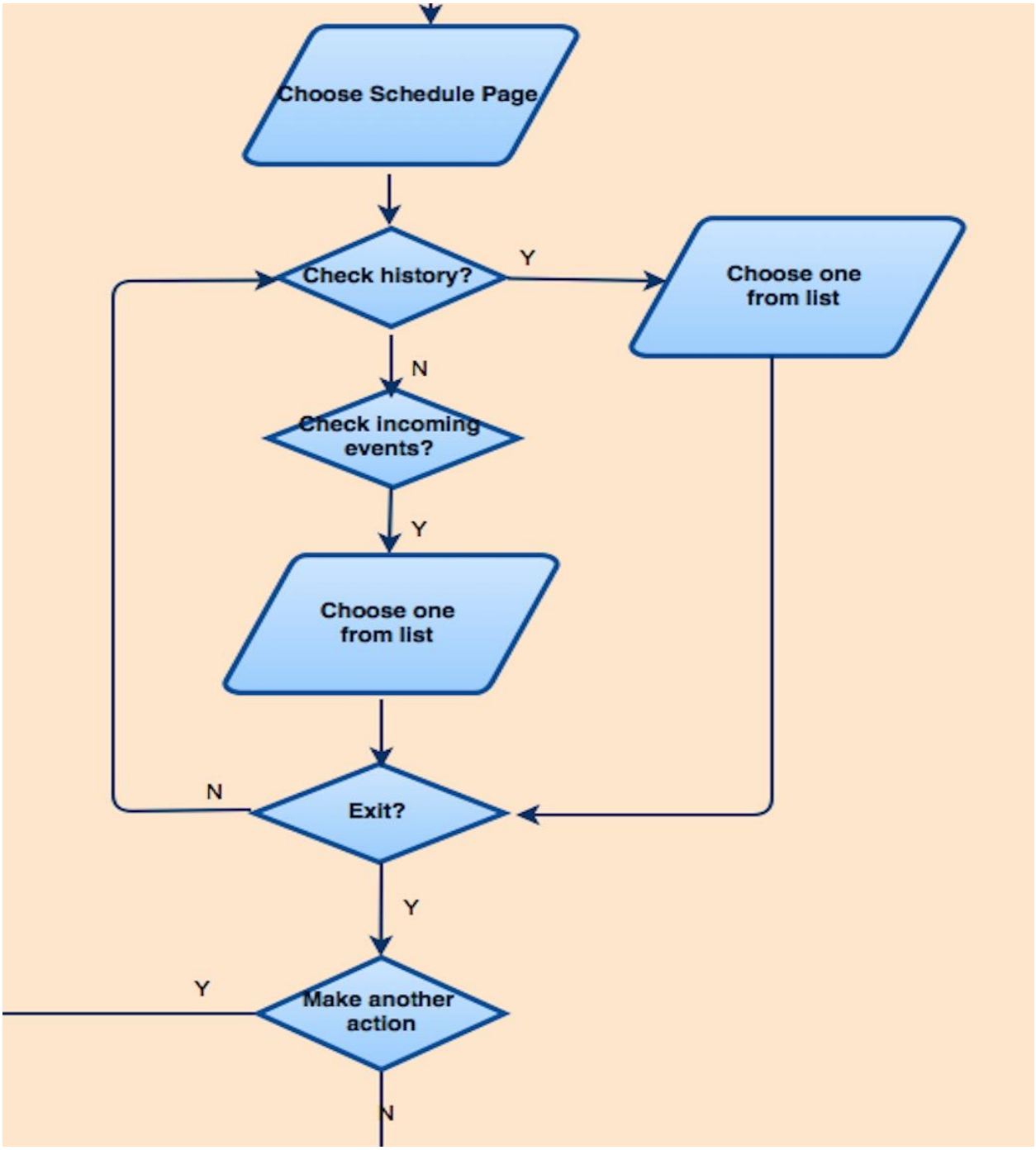
3.4 Friends Page

In this page, user can see all of his/her friends. He/She can choose to chat with one of them, see friends' information and invite them to an activity. After that, user can make another action.



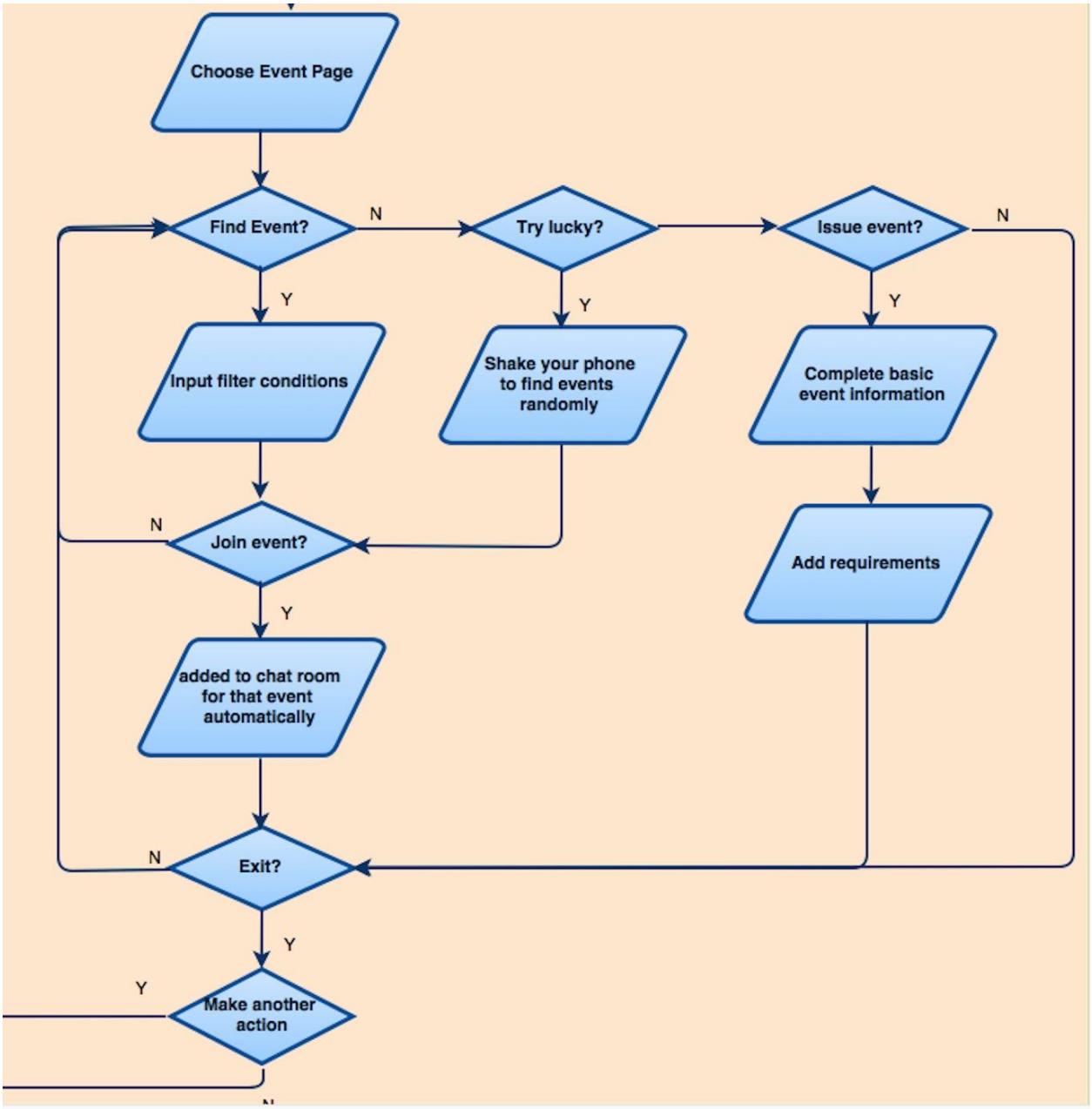
3.5 Schedule Page

In this page, user can choose to list all activities he/she has joined or issued or to see all activities in the format of Date. User can choose one activity and then enter into that event page to see detailed information. After that, user can make another action.



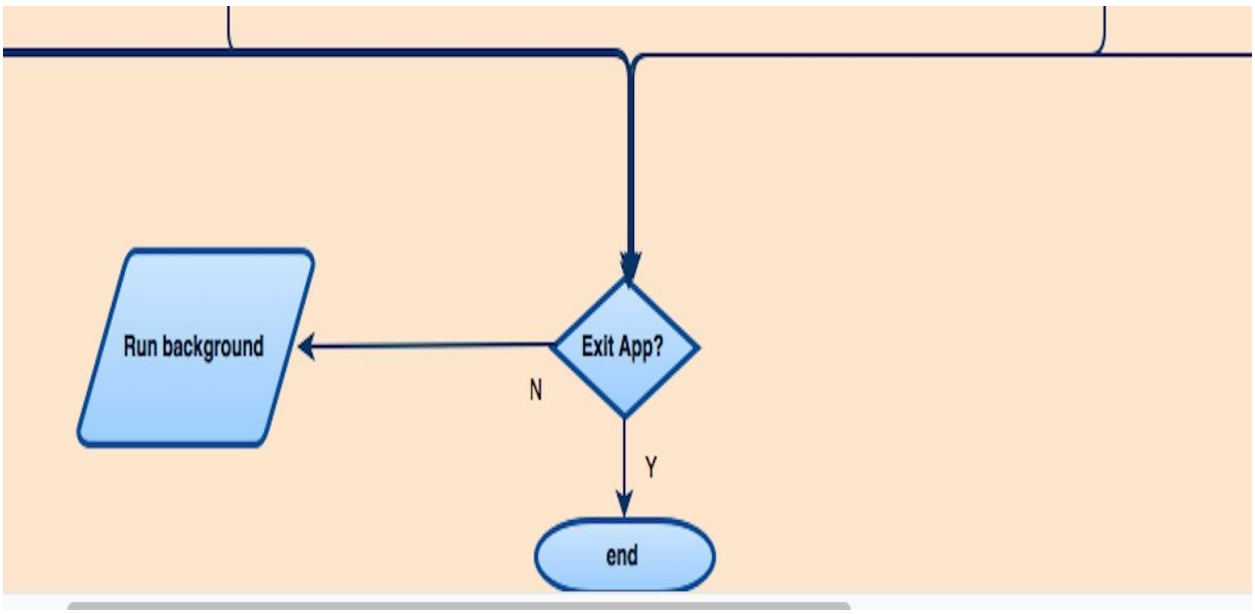
3.6 Event Page

User can enter Event page, where he/she can join or issue an activity. By clicking “Find Event” button, all events will shown in a list. User can add filter conditions to see certain events he/she preferred. By clicking “Issue Event” button, user can hold his/her own event. User can determine the name, location, time and limits. After that, user can make another action.

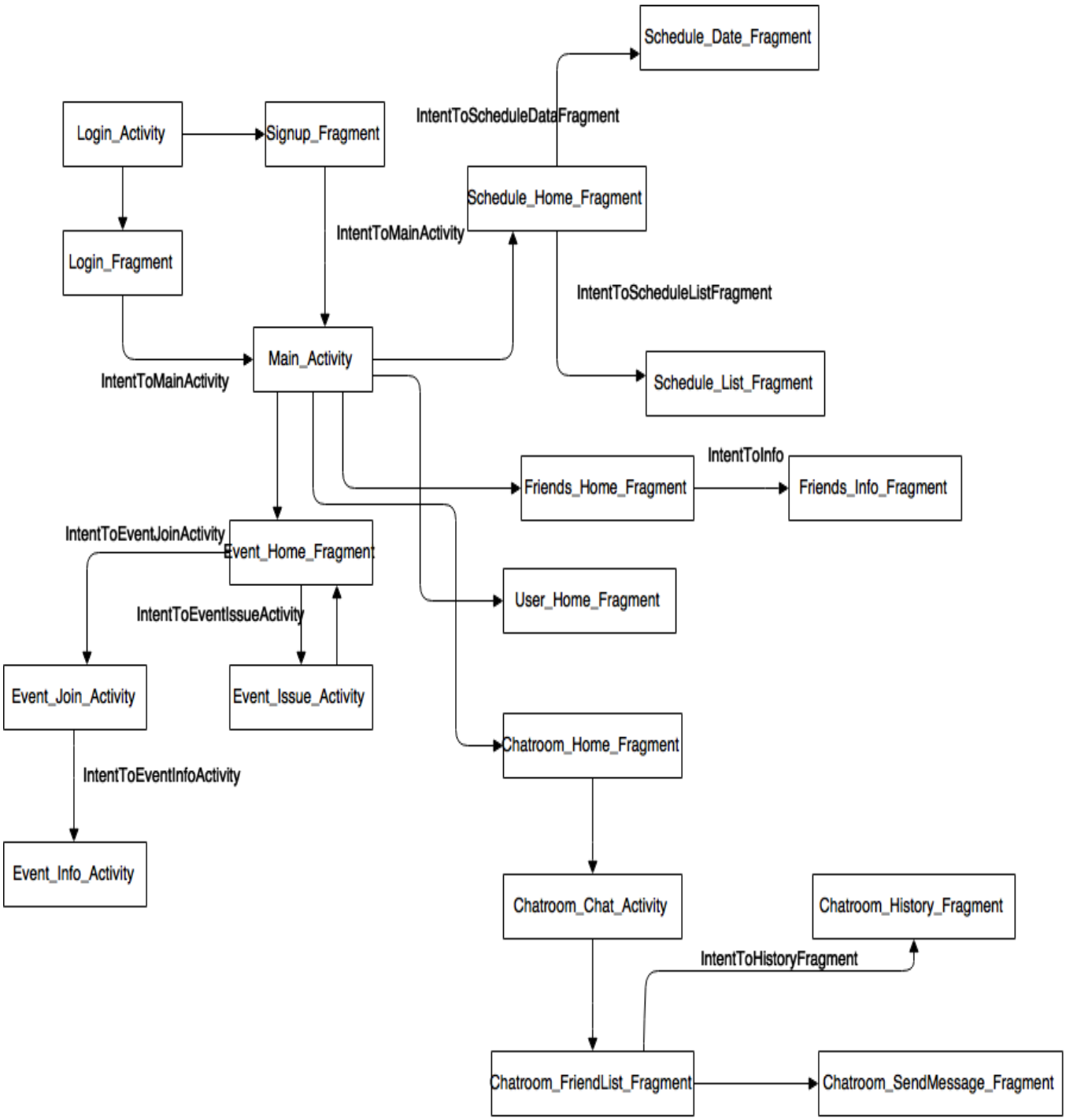


3.7 Stop part

In this part, user can exit BuzzBee application or keep it running in background.



4.PageFlow Diagram

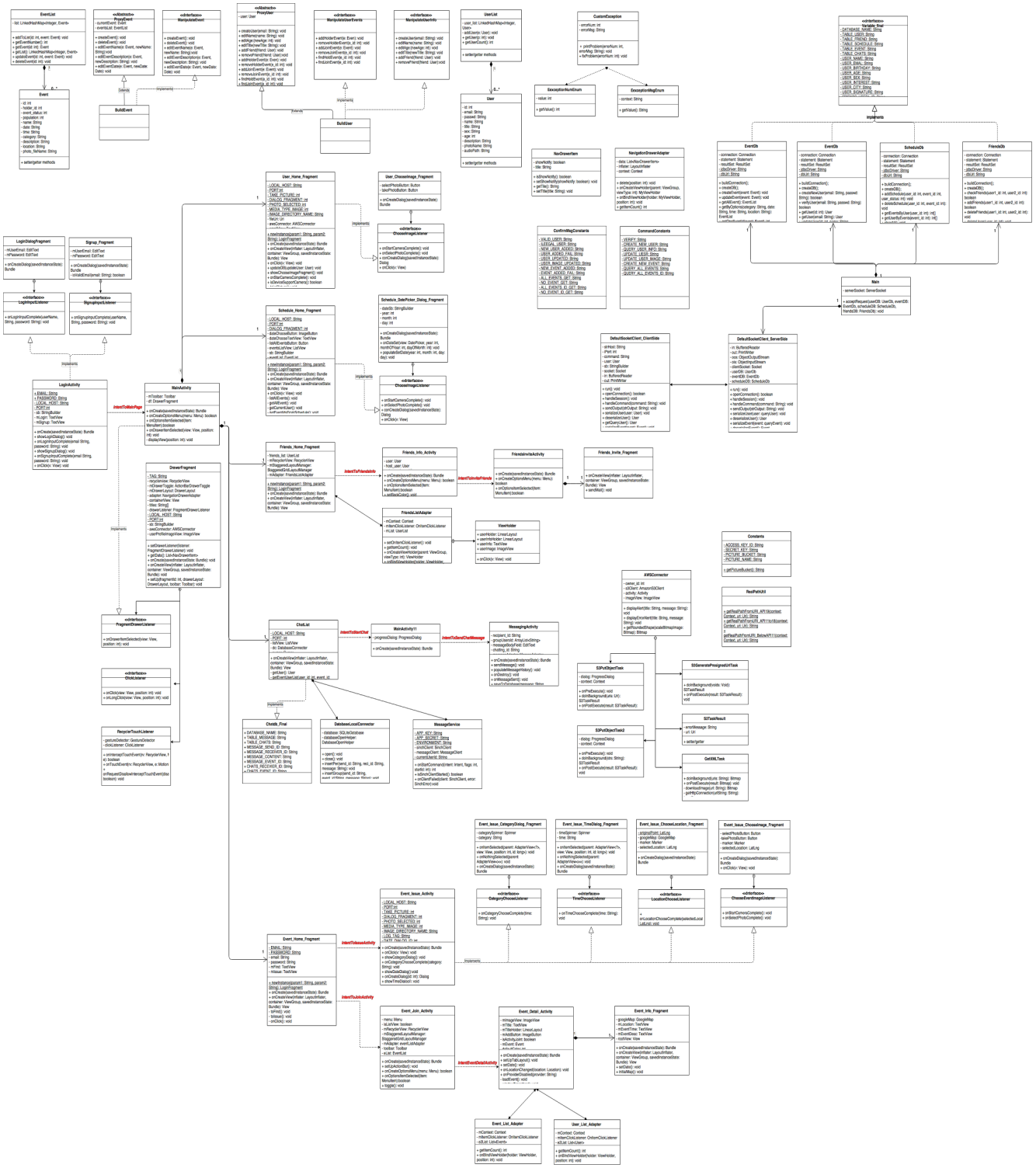


The diagram above shows the relationship between different activities and fragments. Each time we press a button, or type in an input to request, we

may move from one to another. And for here, we could initialize a new Intent object and call a method to complete the jump.

5.Class Diagram

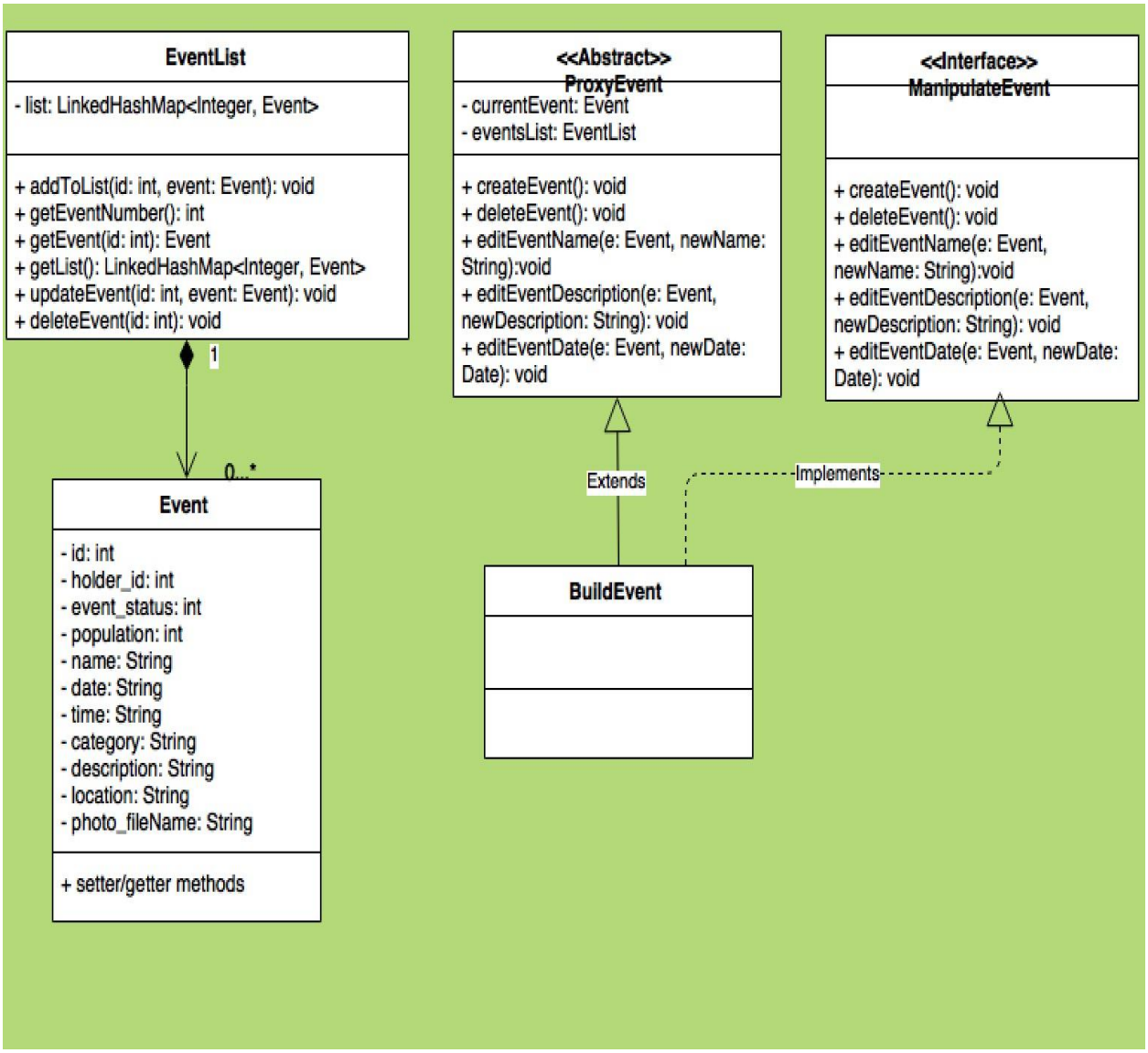
This is the whole picture for class diagram.



Following shows different parts of class diagram.

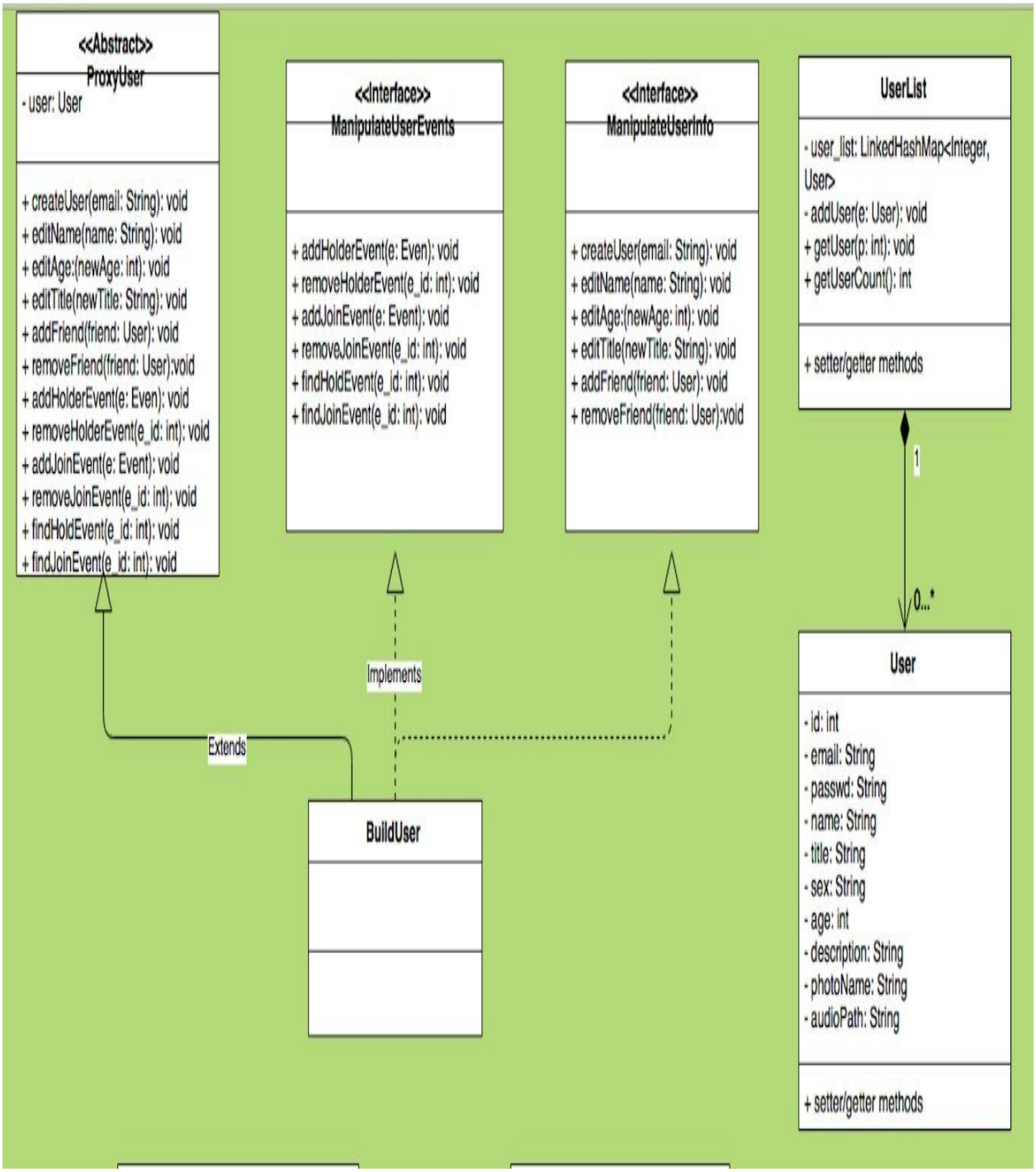
5.1 Event part

This part shows the design for Event Object. We use interface and abstract class for the purpose of encapsulation. BuildEvent is an empty class left for external world access.



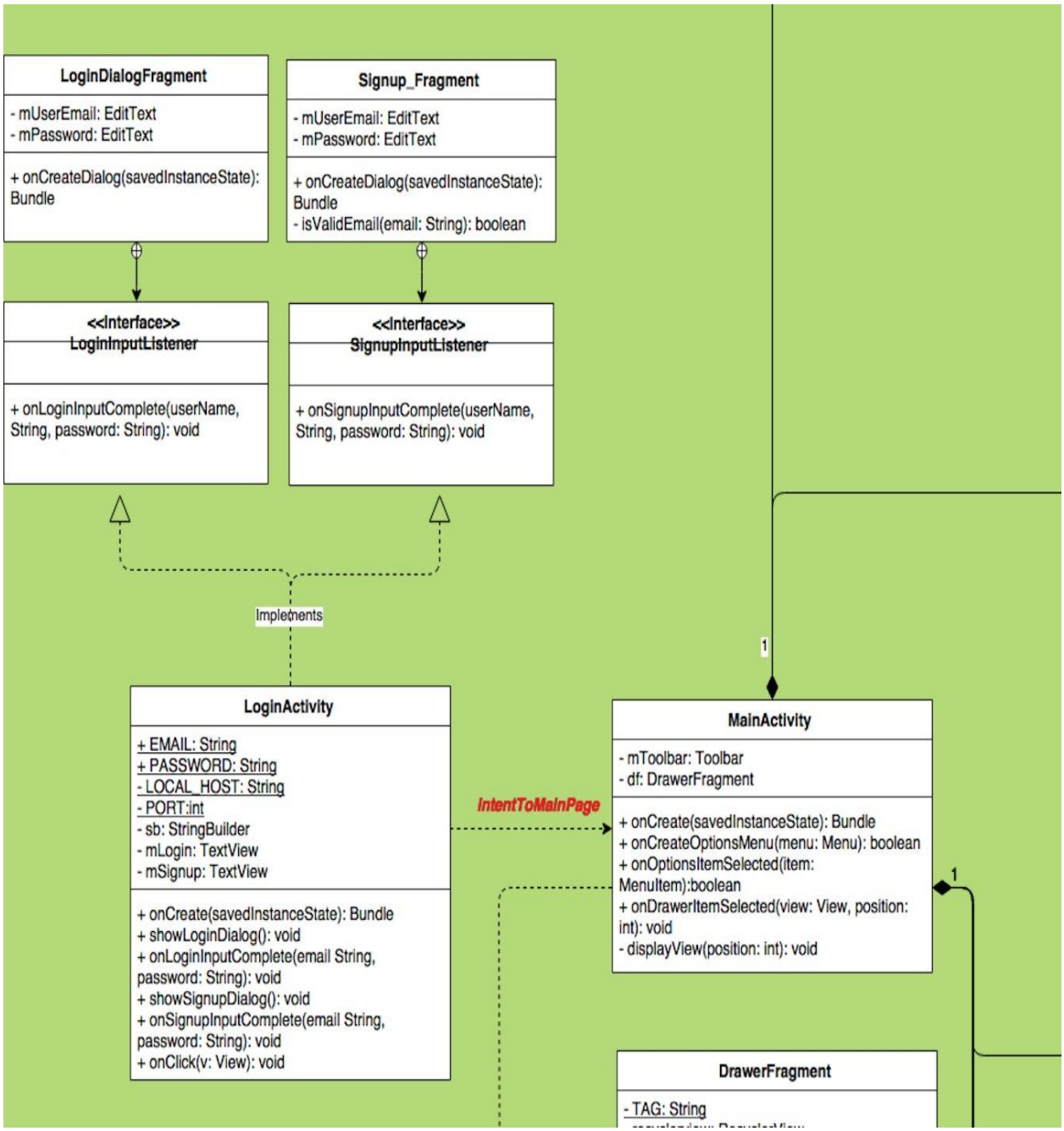
5.2 User part

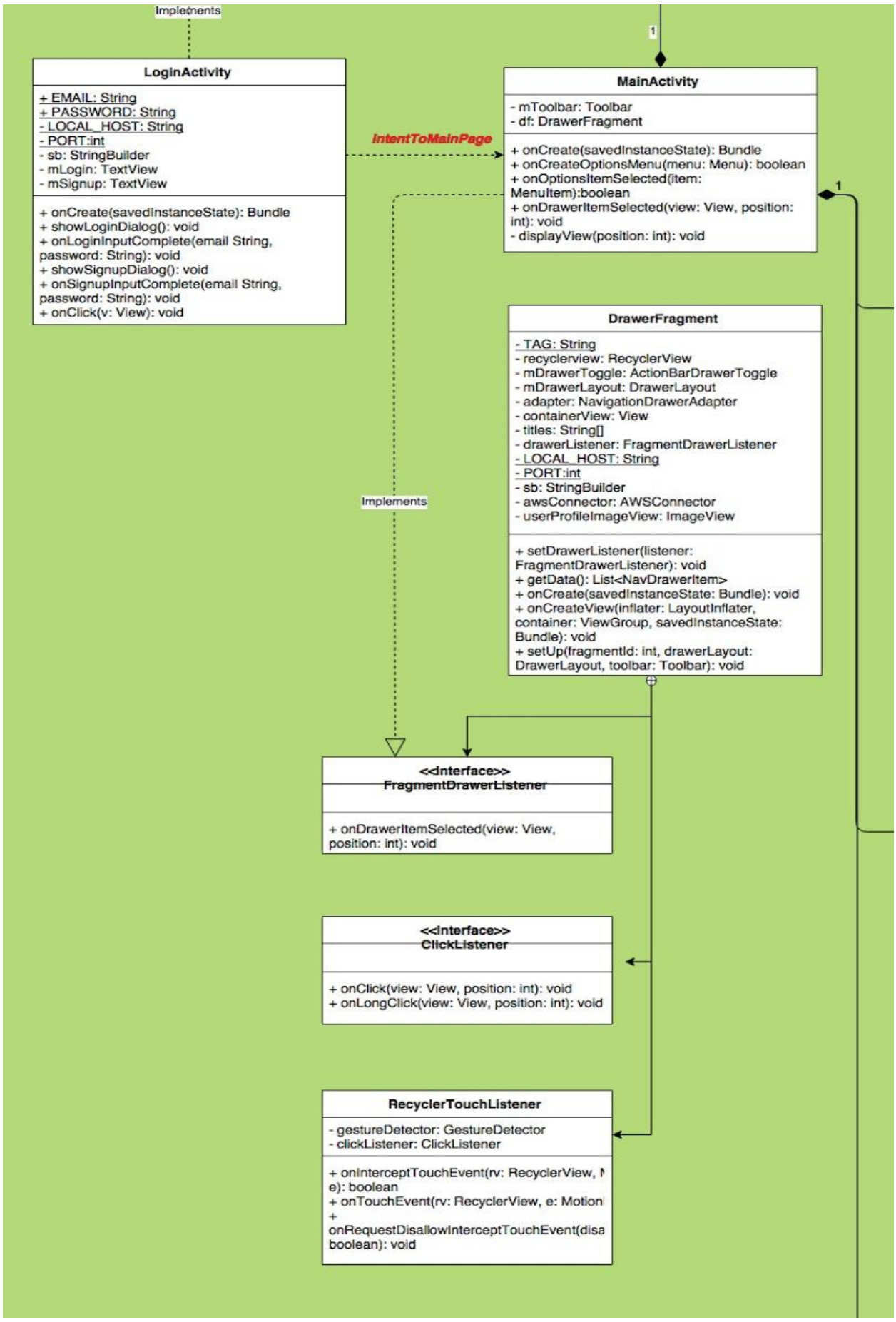
This part shows the design for User Object. Like Event design, we use interface and abstract class for the purpose of encapsulation. BuildUser is an empty class left for external world access.



5.3 LoginActivity → MainActivity

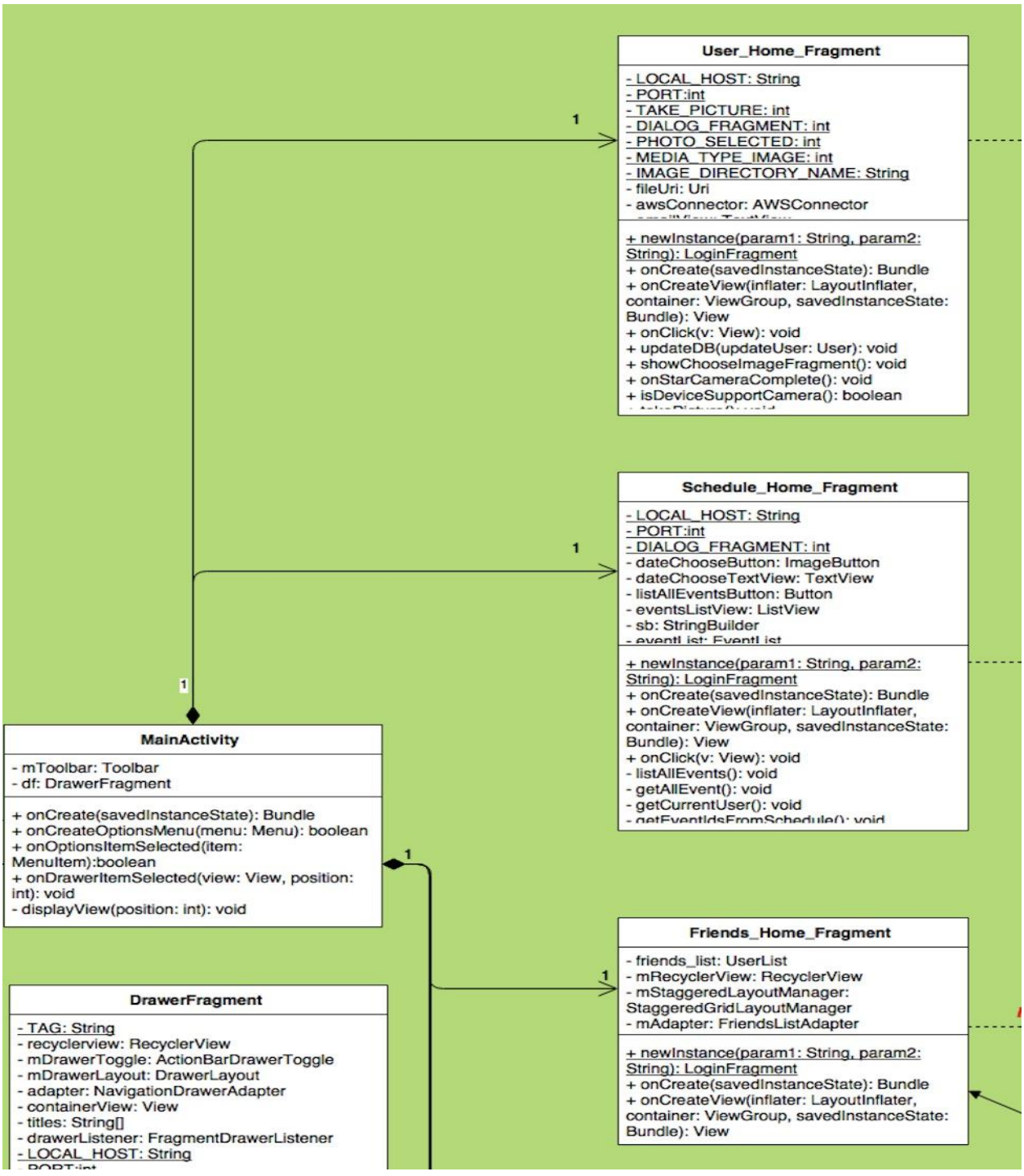
This picture shows the component of LoginActivity, and how to transfer from LoginActivity to MainActivity.

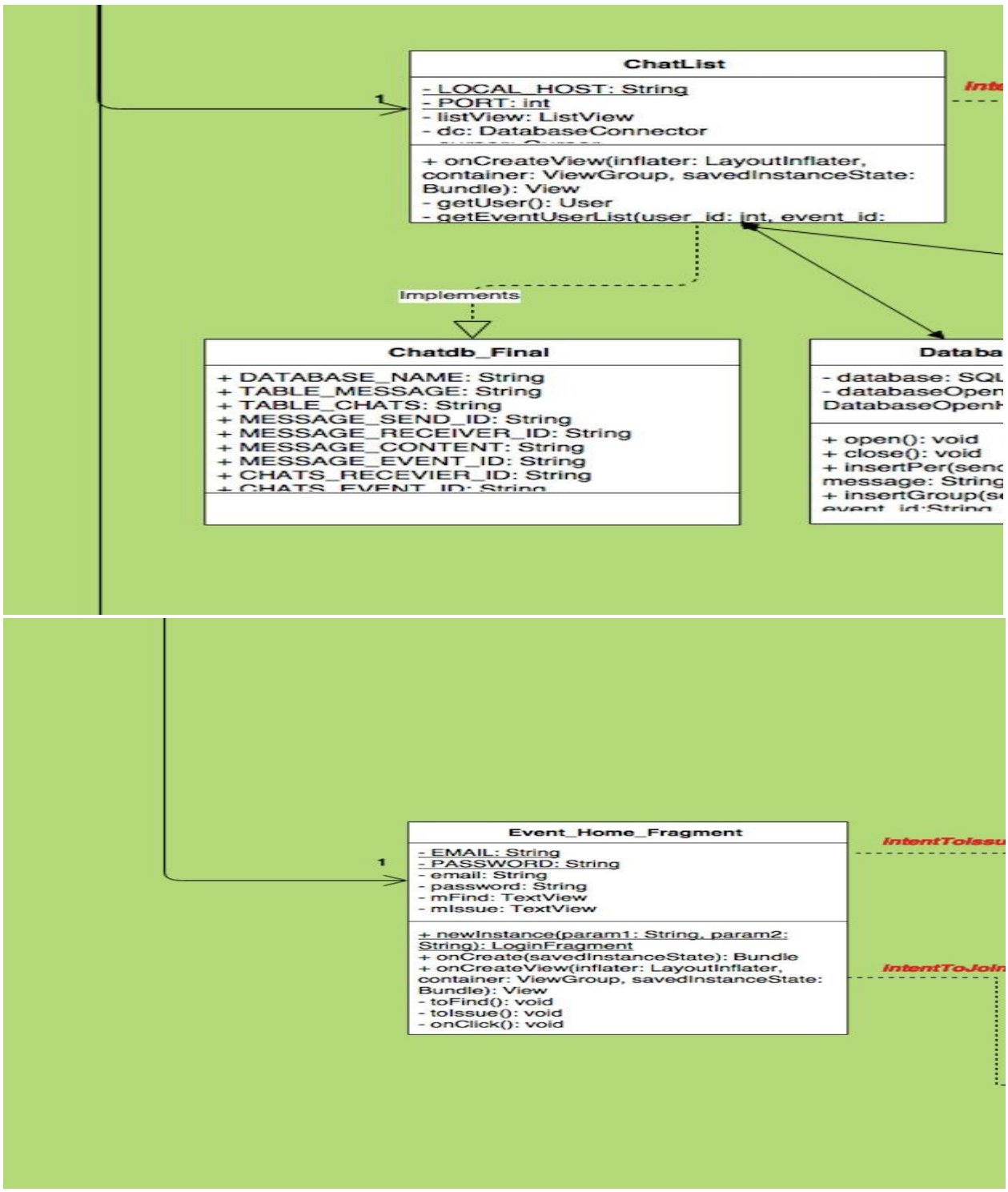



Implements
LoginActivity
+ EMAIL: String
+ PASSWORD: String
- LOCAL_HOST: String
- PORT:int
- sb: StringBuilder
- mLogin: TextView
- mSignup: TextView
+ onCreate(savedInstanceState): Bundle
+ showLoginDialog(): void
+ onLoginInputComplete(email String, password: String): void
+ showSignupDialog(): void
+ onSignupInputComplete(email String, password: String): void
+ onClick(v: View): void
IntentToMainPage
1
MainActivity
- mToolbar: Toolbar
- df: DrawerFragment
+ onCreate(savedInstanceState): Bundle
+ onCreateOptionsMenu(menu: Menu): boolean
+ onOptionsItemSelected(item: MenuItem):boolean
+ onDrawerItemSelected(view: View, position: int): void
- displayView(position: int): void
1
Implements
DrawerFragment
- TAG: String
- recyclerview: RecyclerView
- mDrawerToggle: ActionBarDrawerToggle
- mDrawerLayout: DrawerLayout
- adapter: NavigationDrawerAdapter
- containerView: View
- titles: String[]
- drawerListener: FragmentDrawerListener
- LOCAL_HOST: String
- PORT:int
- sb: StringBuilder
- awsConnector: AWSConnector
- userProfileImageView: ImageView
+ setDrawerListener(listener: FragmentDrawerListener): void
+ getData(): List<NavDrawerItem>
+ onCreate(savedInstanceState: Bundle): void
+ onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): void
+ setUp(fragmentId: int, drawerLayout: DrawerLayout, toolbar: Toolbar): void
<<Interface>>
FragmentDrawerListener
+ onDrawerItemSelected(view: View, position: int): void
<<Interface>>
ClickListener
+ onClick(view: View, position: int): void
+ onLongClick(view: View, position: int): void
RecyclerTouchListener
- gestureDetector: GestureDetector
- clickListener: ClickListener
+ onInterceptTouchEvent(rv: RecyclerView, N
e): boolean
+ onTouchEvent(rv: RecyclerView, e: Motion
+
onRequestDisallowInterceptTouchEvent(disa
boolean): void

5.4 Main Page

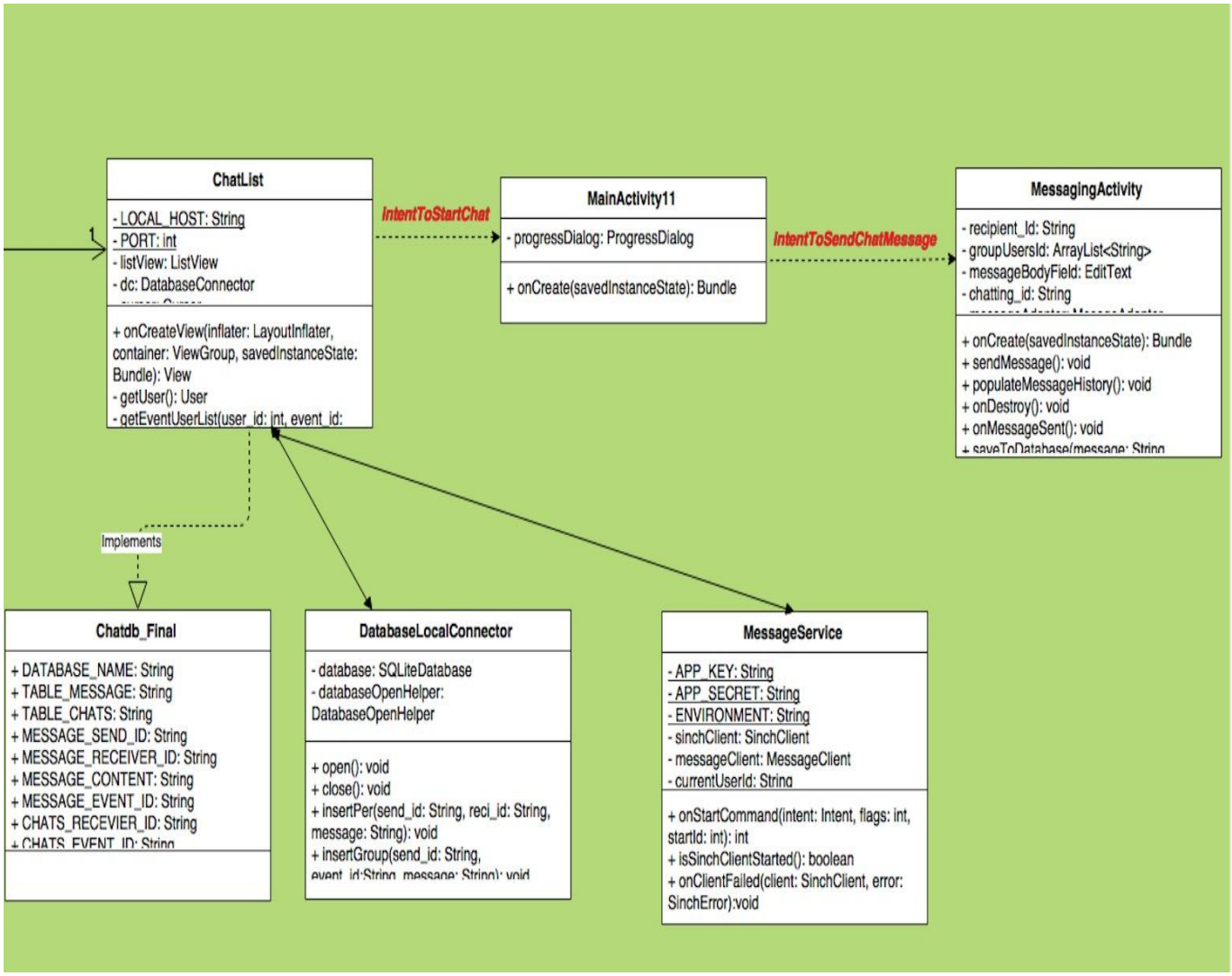
This pictures shows the five fragments which compose MainActivity. User can transfer among those five fragments





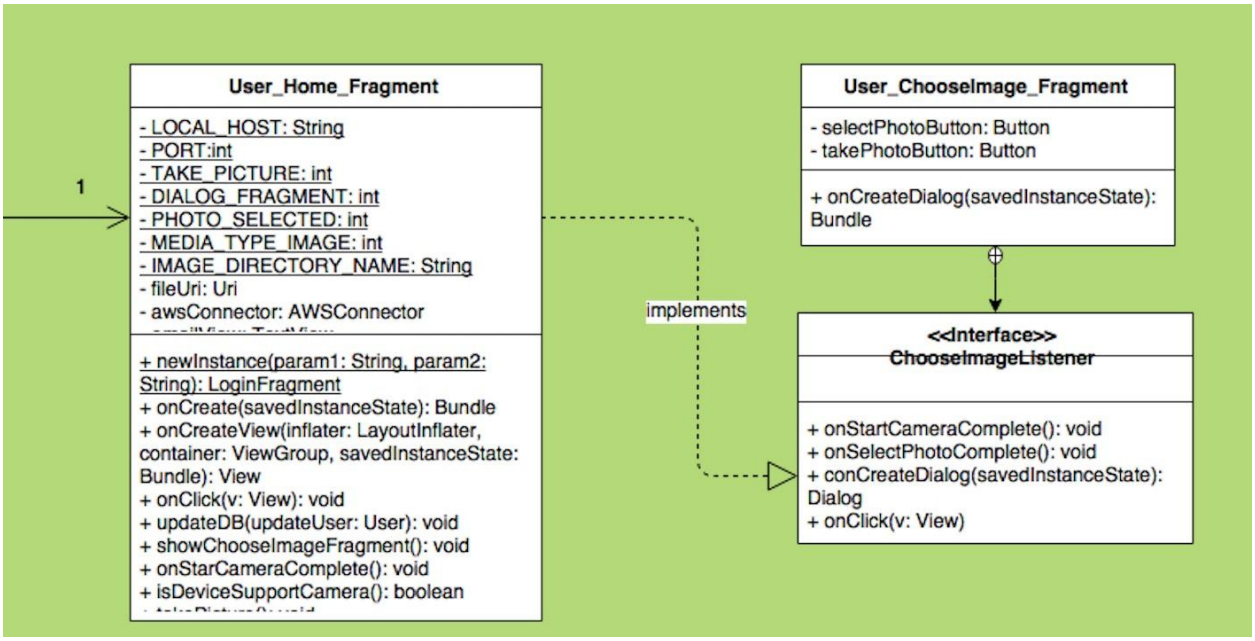
5.5 Chat room part

There shows the first fragment that compose MainActivity. It contains 3 fragments.



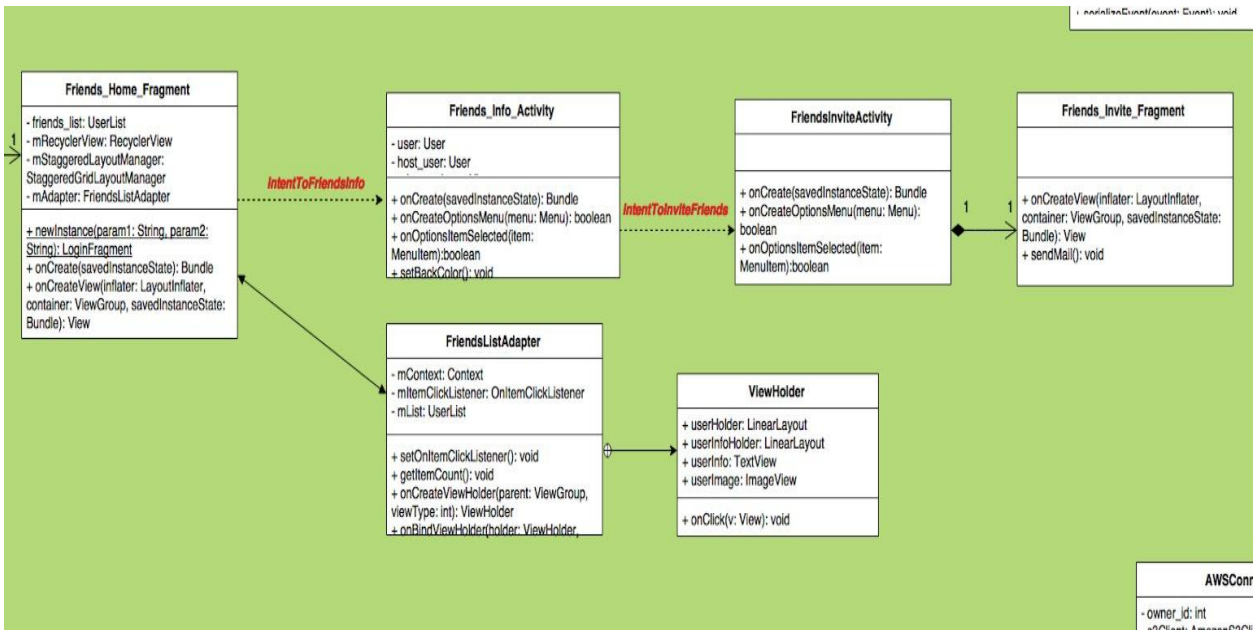
5.6 User Info part

There shows User_Home_Fragment, the second fragment that compose MainActivity, in which user can view or edit personal information.



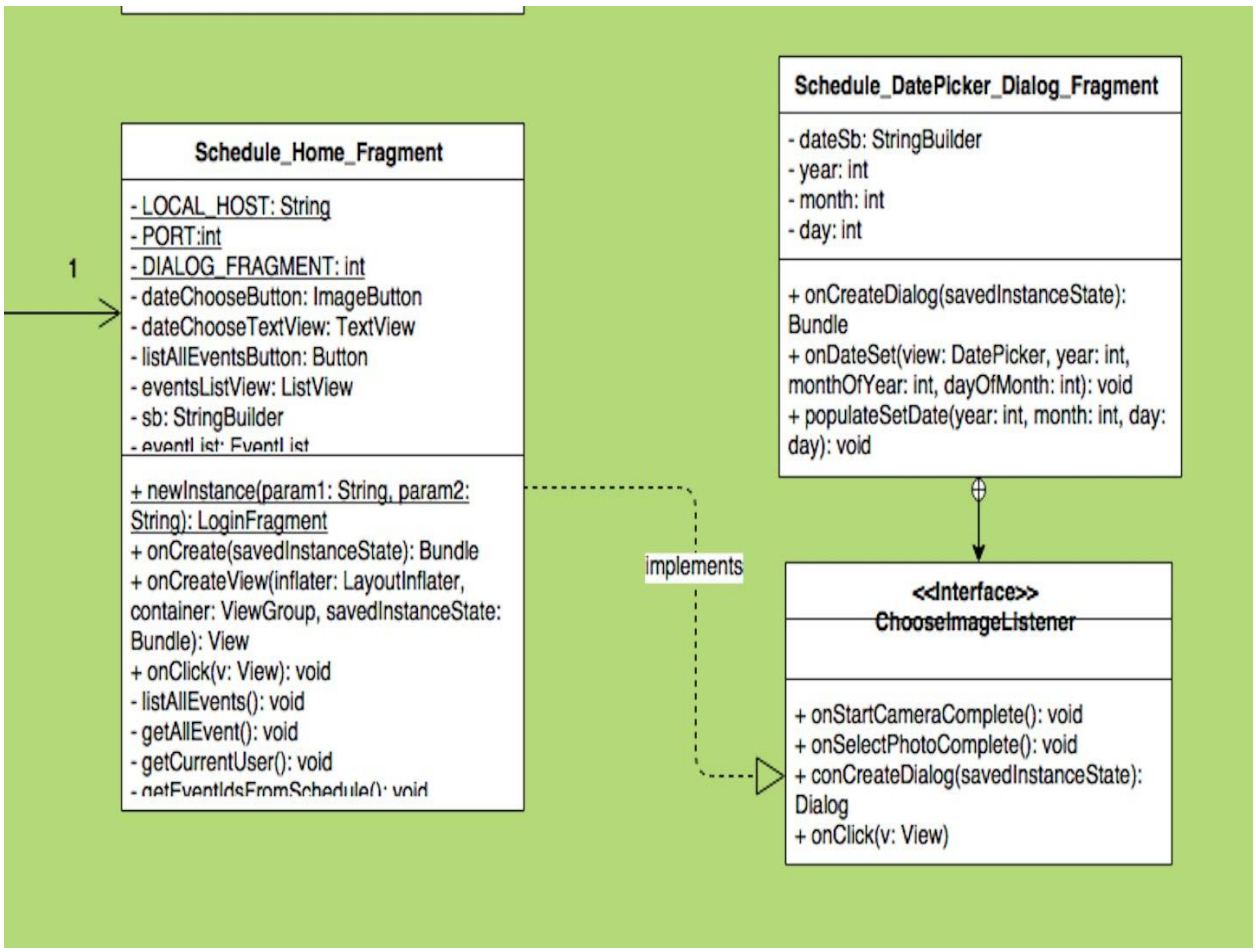
5.7 Friends part

The third fragment that compose MainActivity allows user to view friends information and chat with them.



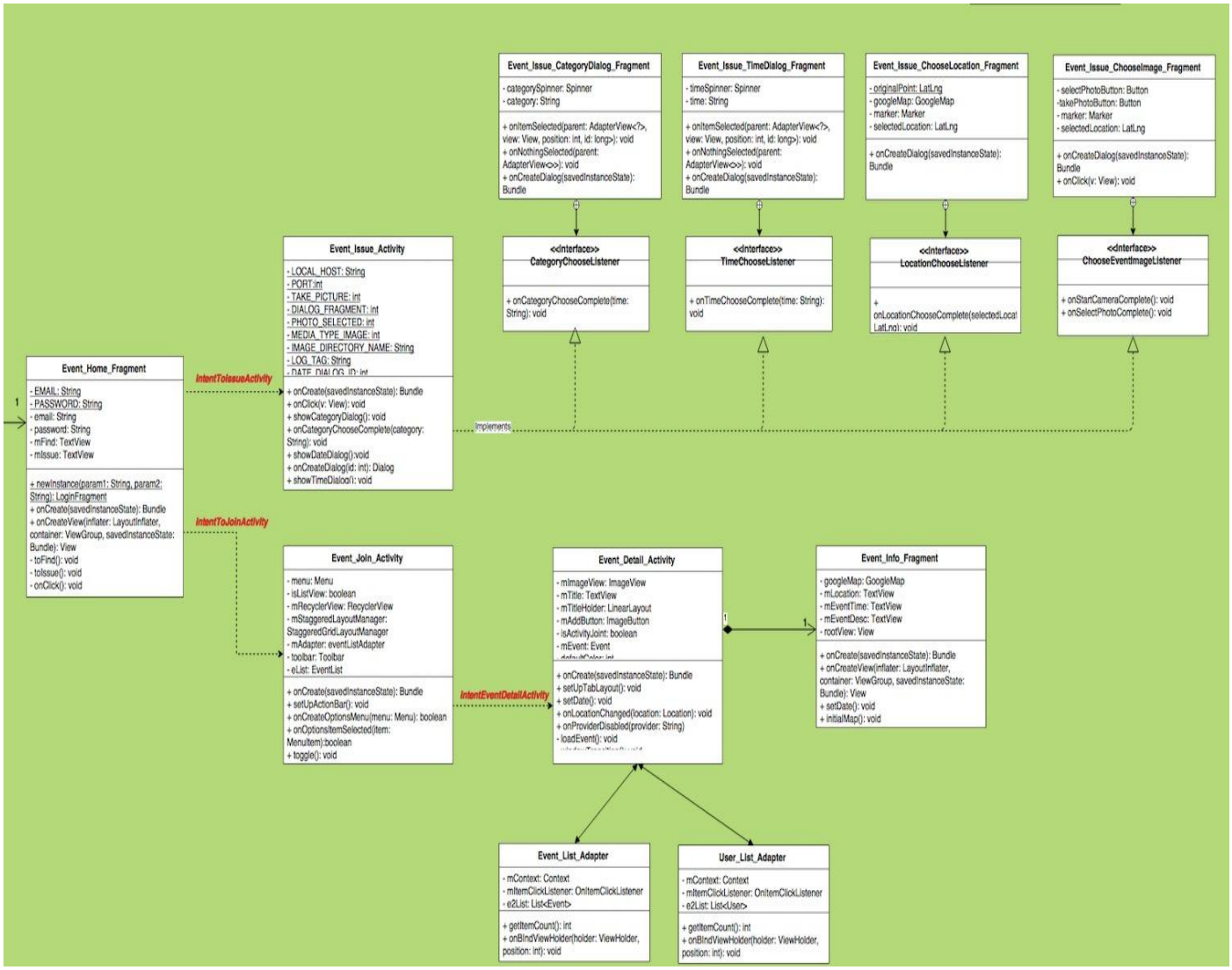
5.8 Schedule part

The fourth fragment allows user to view all activities he/she has joined or issued. DatePicker is used in this part.



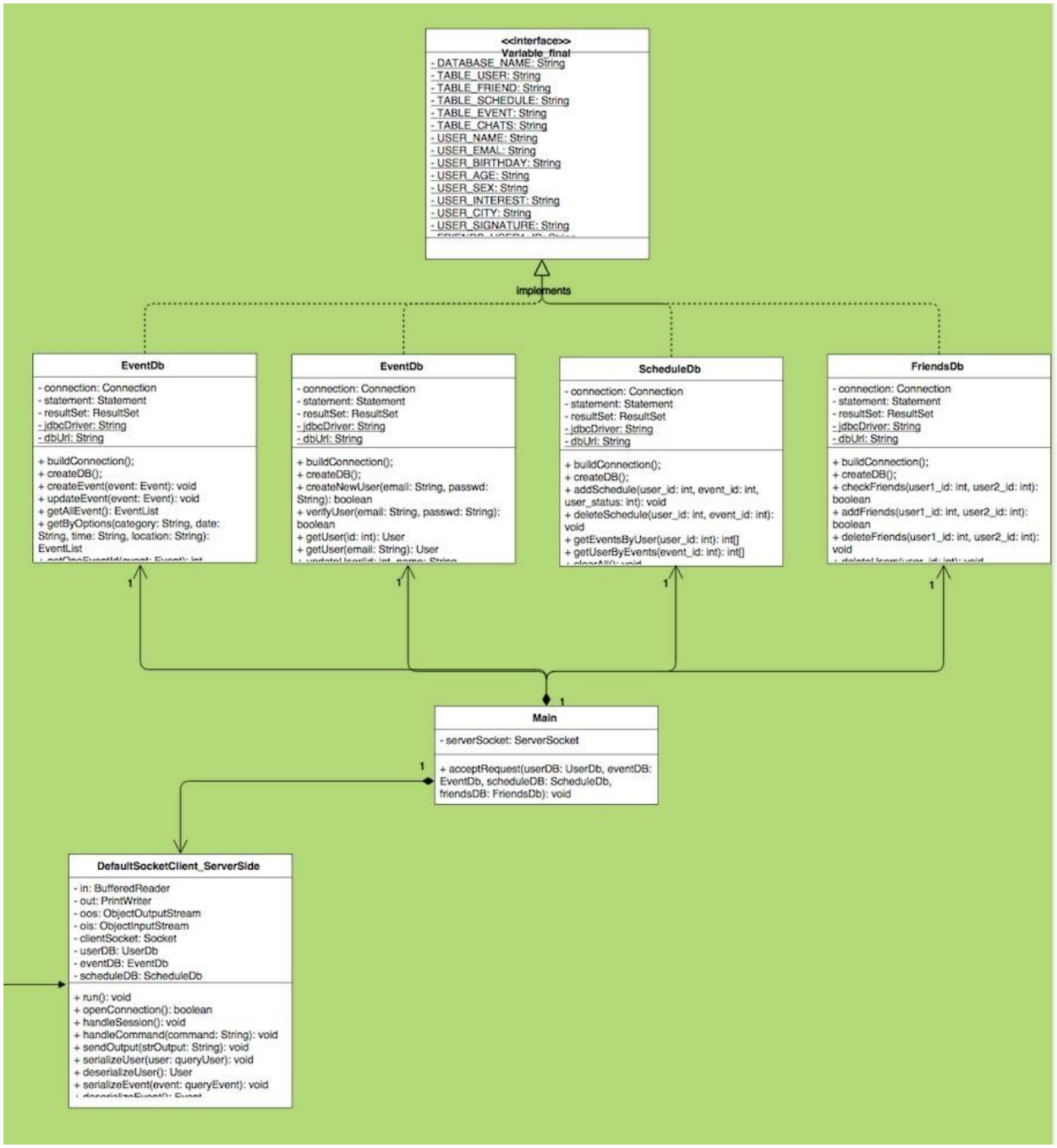
5.9 Event part

In this part, user can issue or join an activity. To join an activity, system will list all the activities that a user can join in. User can also set filter conditions to only view the ones he/she prefers. To issue an activity, user need to specify the category, name, location, date, limits and etc. information.

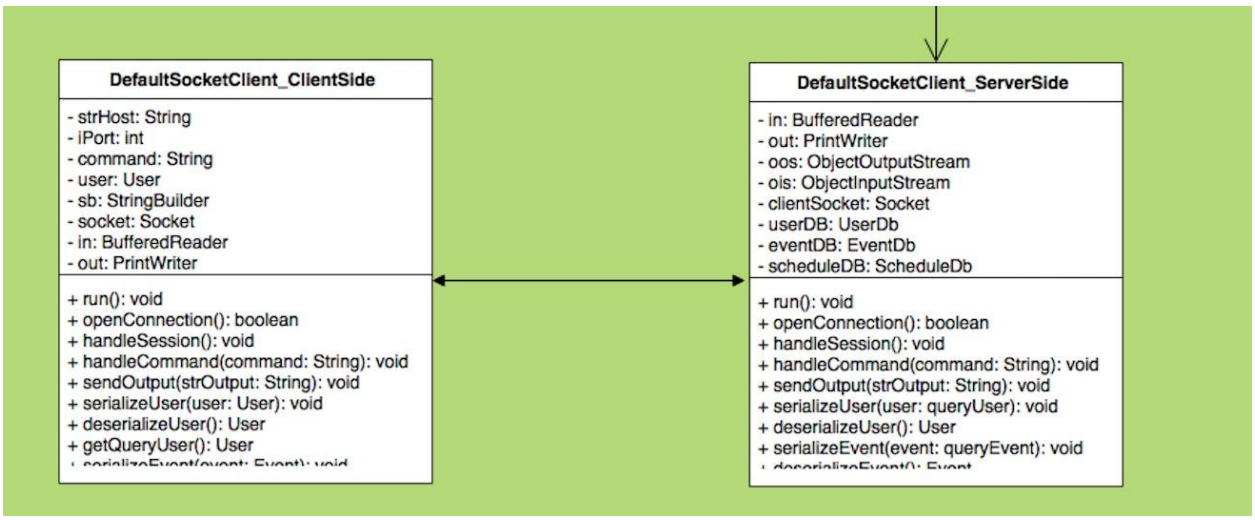


5.10 database part

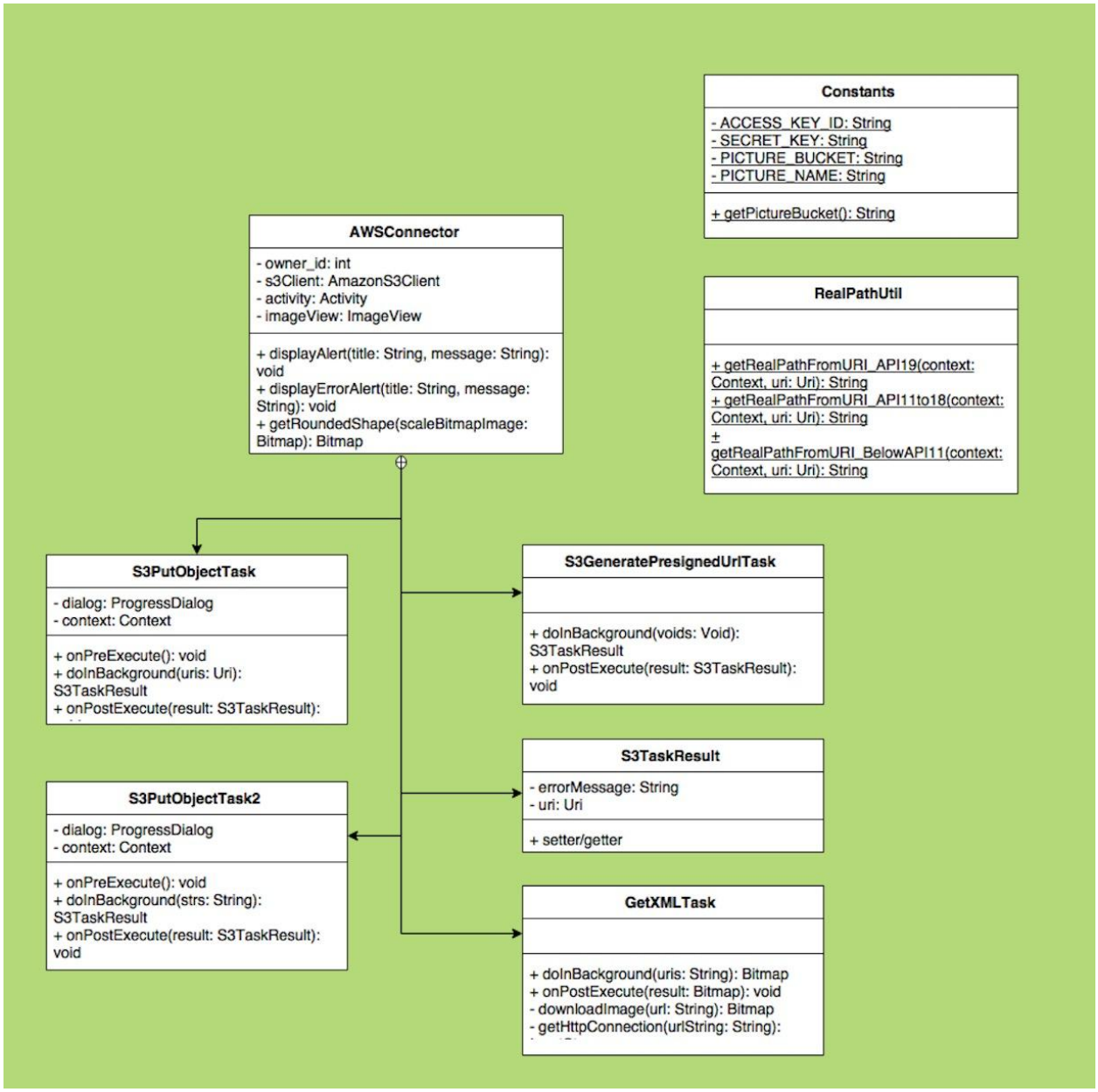
Database can be used to save and get data. There is one remote database, MySQL which is used to save user and event information, and one local database, which is used to save chatting message.



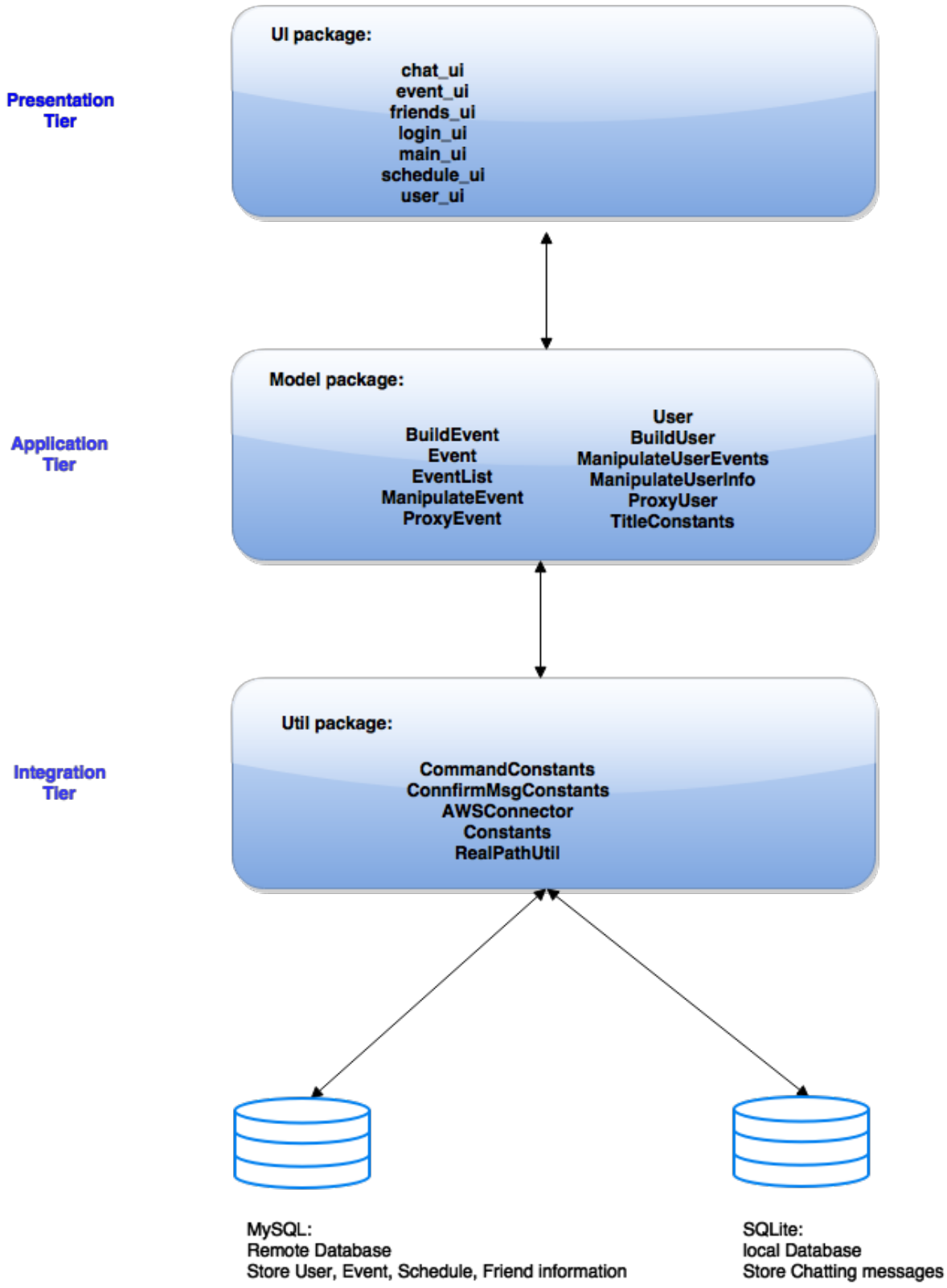
5.11 socket part


DefaultSocketClient_ClientSide
- strHost: String
- iPort: int
- command: String
- user: User
- sb: StringBuilder
- socket: Socket
- in: BufferedReader
- out: PrintWriter
+ run(): void
+ openConnection(): boolean
+ handleSession(): void
+ handleCommand(command: String): void
+ sendOutput(strOutput: String): void
+ serializeUser(user: User): void
+ deserializeUser(): User
+ getQueryUser(): User
DefaultSocketClient_ServerSide
- in: BufferedReader
- out: PrintWriter
- oos: ObjectOutputStream
- ois: ObjectInputStream
- clientSocket: Socket
- userDB: UserDb
- eventDB: EventDb
- scheduleDB: ScheduleDb
+ run(): void
+ openConnection(): boolean
+ handleSession(): void
+ handleCommand(command: String): void
+ sendOutput(strOutput: String): void
+ serializeUser(user: queryUser): void
+ deserializeUser(): User
+ serializeEvent(event: queryEvent): void

5.12 Amazon S3 part



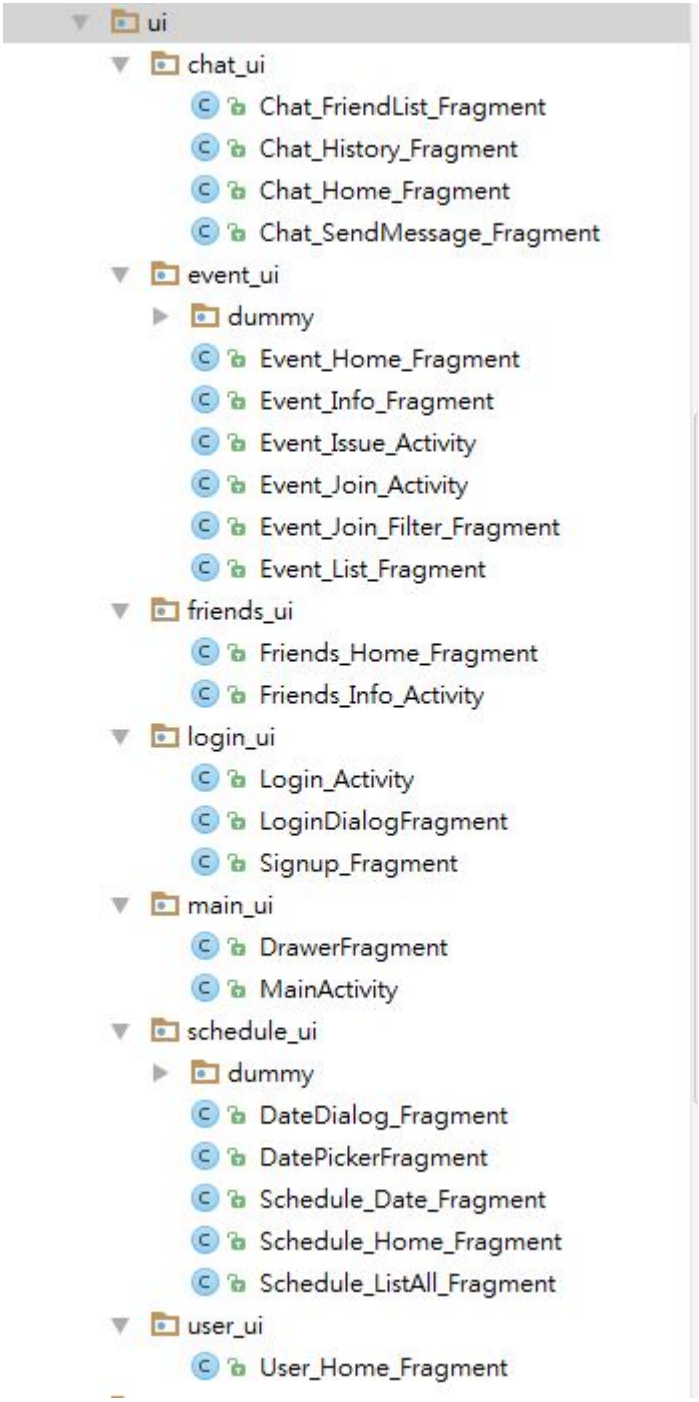
6.Tiers Interaction Diagram



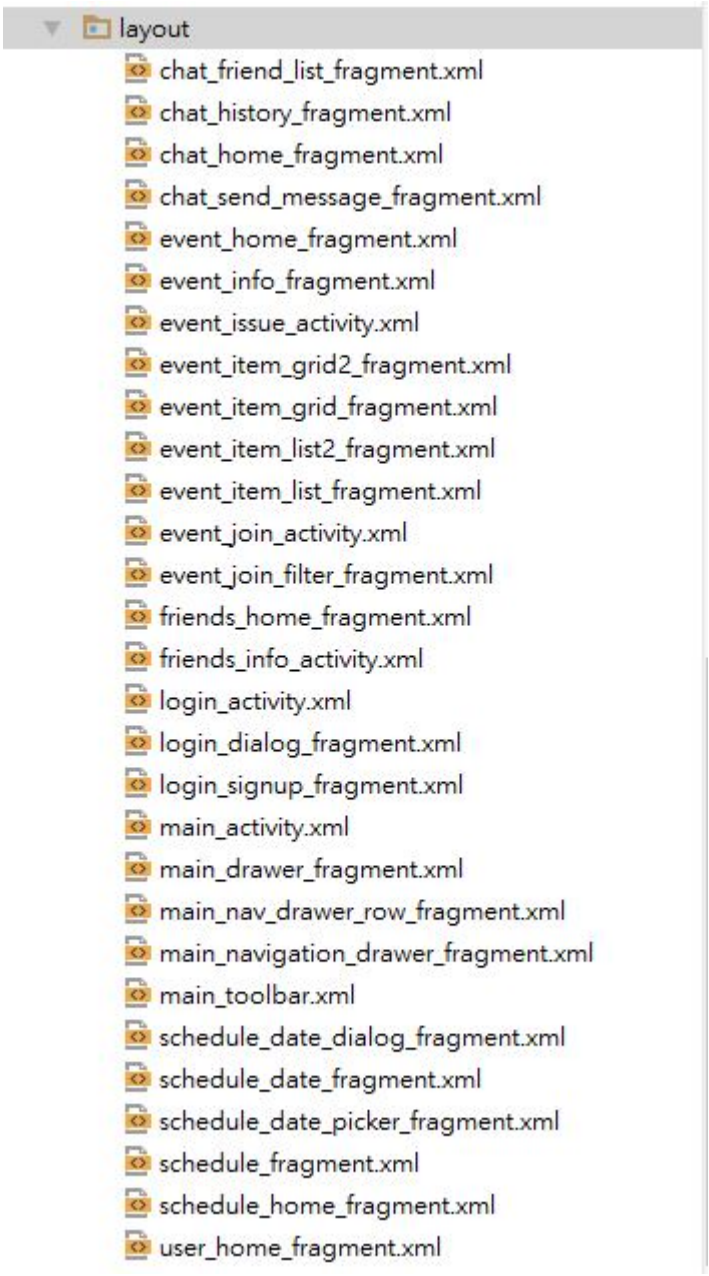
7.Designing Presentation Tier

With BuzzBee, user can choose to enter chat room, check personal information, view friends, check schedule and join/issue events. For all these functions, we separate each to a different package.
The relationship between intents and activities or fragments has been shown in the fourth part of this design documentation.
(Please refer to -- 4.PageFlow Diagram)

There shows the picture of the presentation tier folder structure.

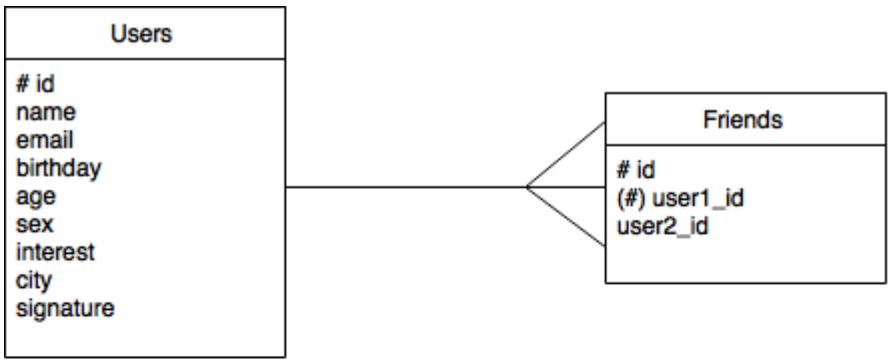


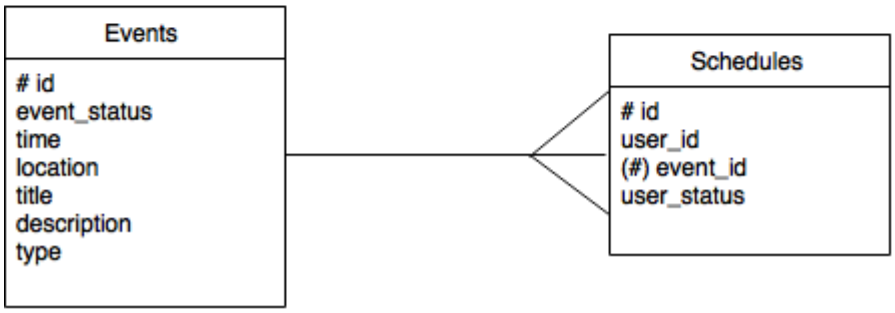
The following picture shows corresponding .xml file.
Each fragment comply with the naming rule: PackaName_ActivityName.xml



8. Designing Content Provider(For Storage)

To save and get all the values conveniently, there are 2 databases which will be created. One is on MySQL called BuzzBee which is remote, and it contains four tables, ‘users’, ‘friends’, ‘schedules’ and ‘events’. The relationship of them is listed below:





As it is listed above, the primary key of them are all their id. The relationship between Users and Friends are one to many, which means that a row of data in Users links to multiple rows in Friends. ‘Users’ table has many columns which are listed, and they are all the attributes of one person. The foreign key (user1_id) of Friends is the primary key of Users. Each time one user makes friends with another, one line of data will be added into ‘friends’ table. ‘user1_id’ and ‘user2_id’ are user_id of the two people.

That’s the same of Events and Schedules. One event has many attributes, and when one user creates an event all the values are initialized. One row of data in Events links to many rows in Schedule. And the foreign key of Schedules(event_id) is the primary key of Events, which one line of data in Events can be accessed through Schedules.

What's more, we will create four classes for creating, reading, updating and deleting in /src/util folder.

The other database is created in SQLite called 'BuzzBee' which is local. This database is mainly used to save all the messages. So there is only one table called 'chatting'. Details are listed below:

The primary key is its id. And when one user chats with another or in a group, a chat_id is created for this dialogue. And each time one user makes a chatting, all these values are saved and one line of data can be created and saved into the database.

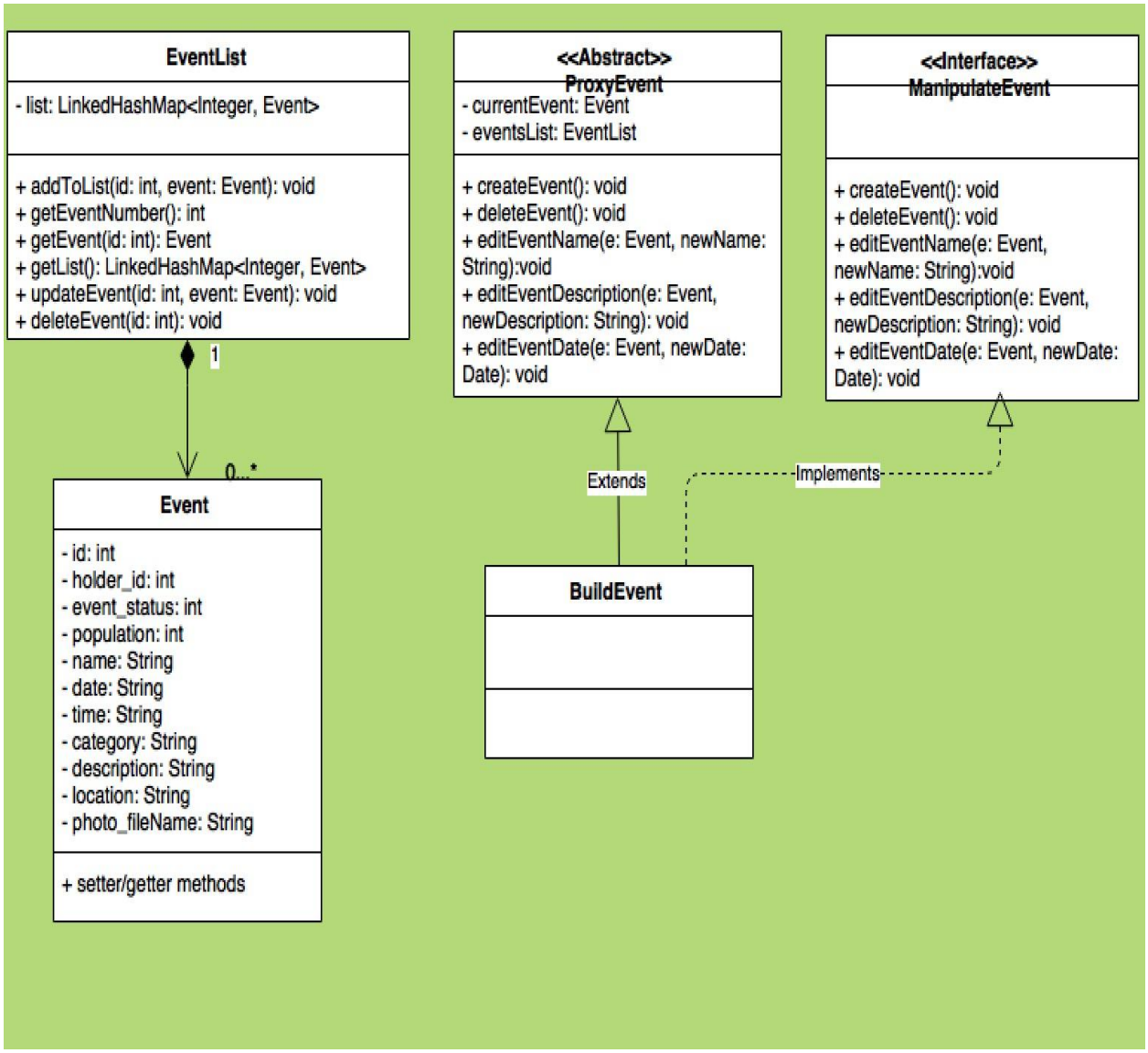
For this database, we will create a DatabaseLocalConnector class in /src/util folder. There will also be inserting methods, deleting methods and reading methods in it.

9.Designing Application Tier(For business logic)

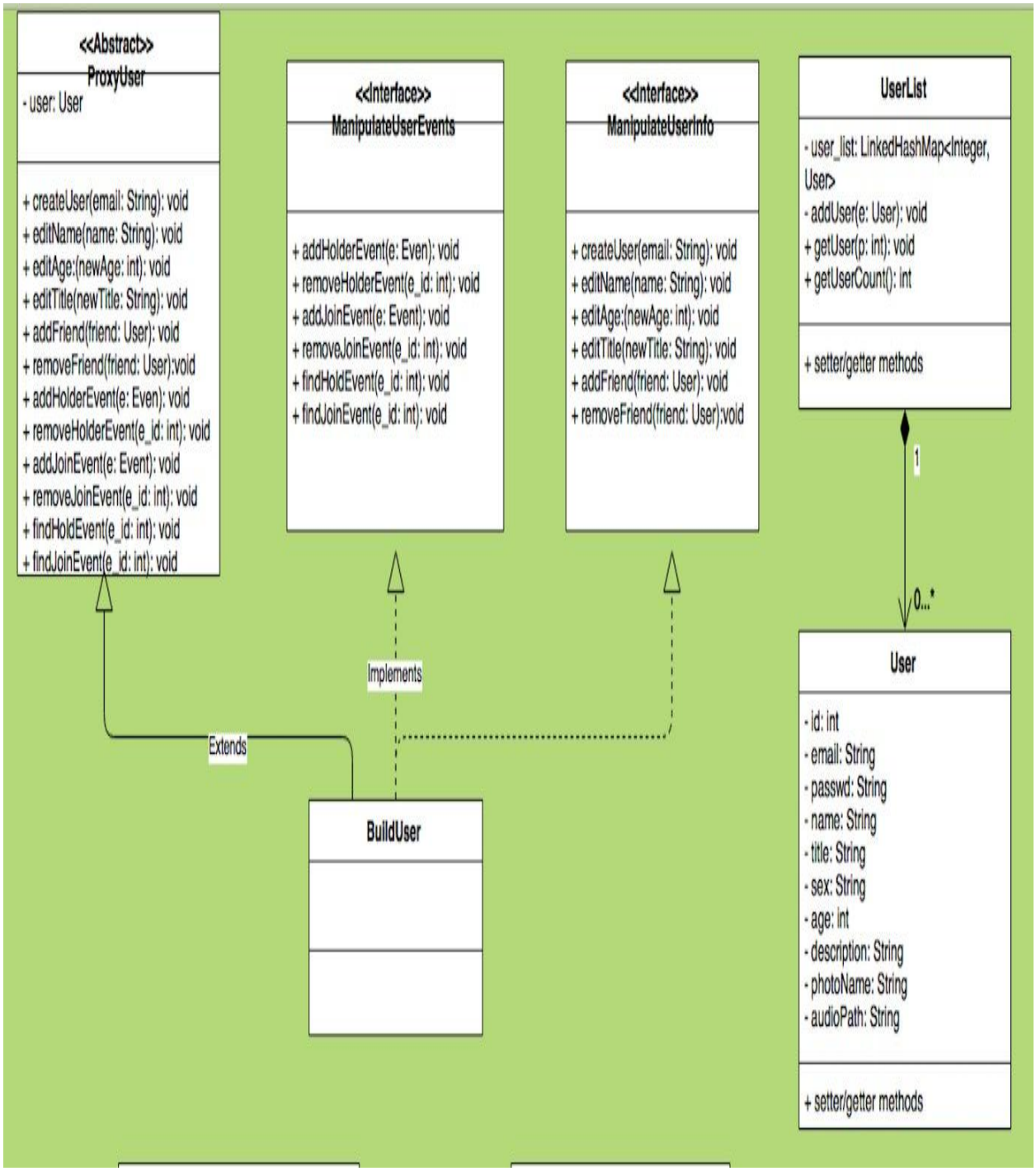
For our android application, we decide to set two models: Event and User.

We use lessons learnt from Mini1 for this design decision. Interface and abstract class are also used for encapsulation purpose. BuildEvent and BuildUser are empty classes. In other words, all methods are implemented in ProxyEvent and ProxyUser classes. So we expose an empty class to external world and hide all implementation details.

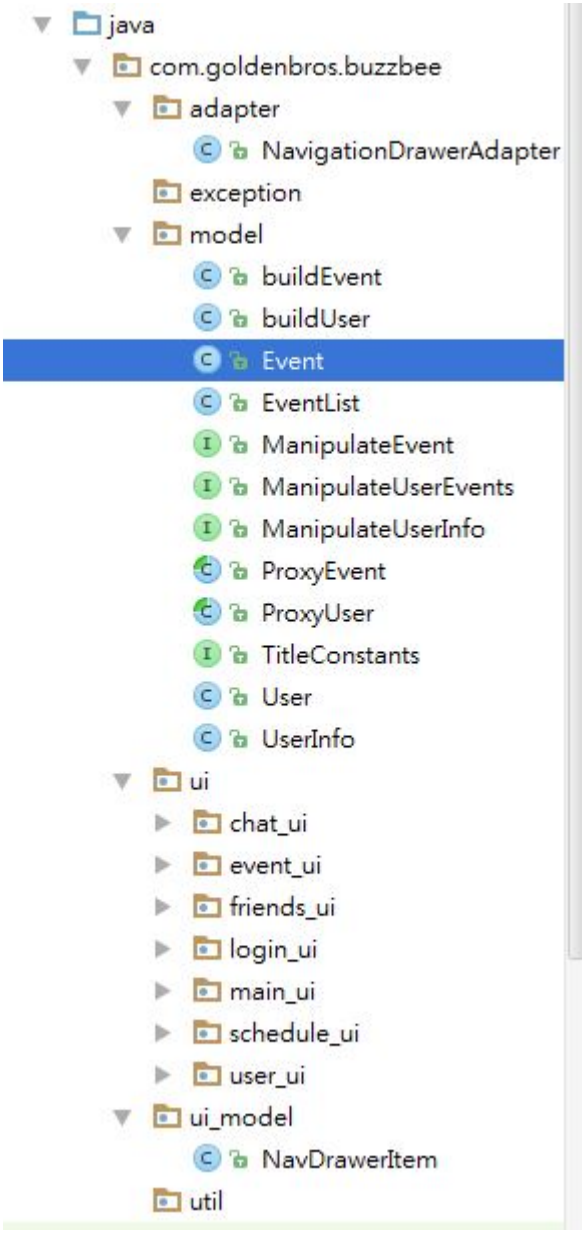
The following picture shows design for Event model.



The following picture shows design for User model.

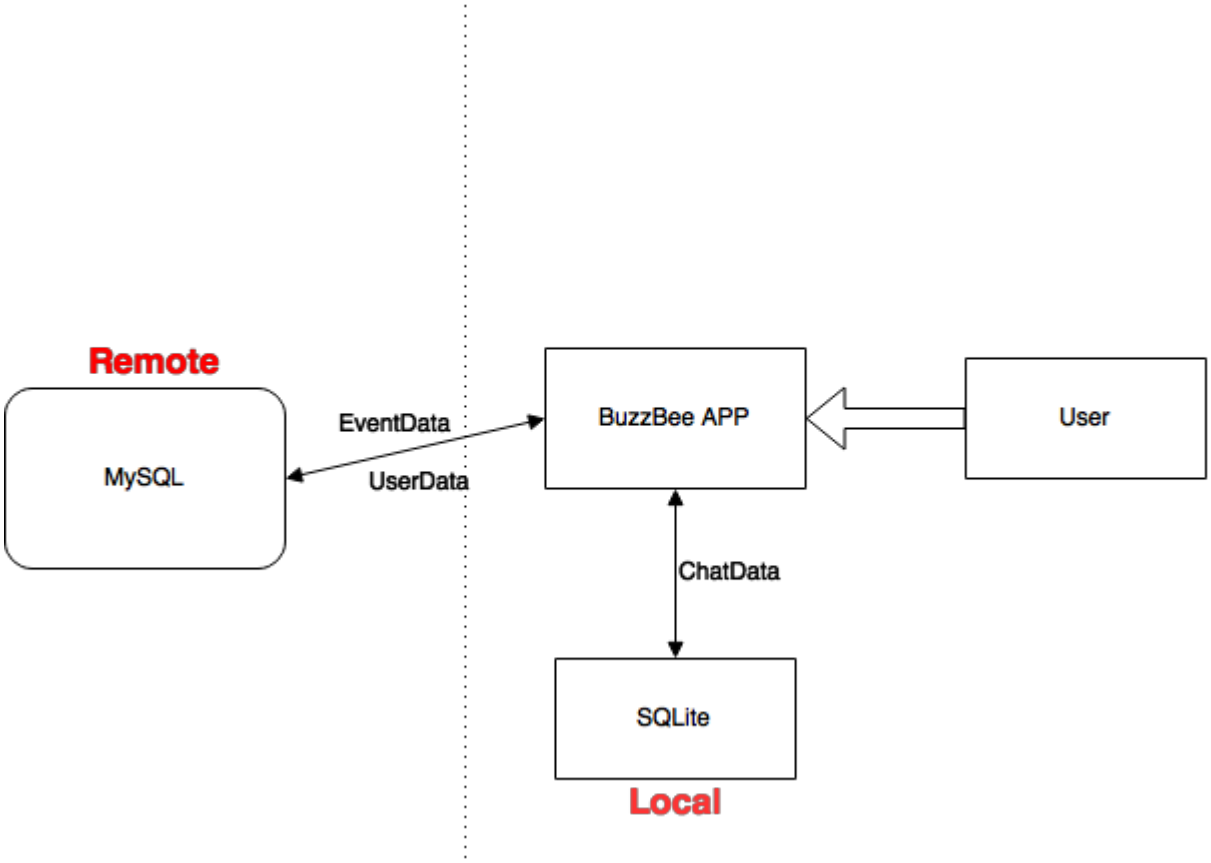


Following shows the folder structure of Model Package.



10. Designing Integration Tier

There will be data interaction between client and server. When users make some requests, the APP will deal with this request and request data from the server in the backend. The interaction diagram is listed below:



When new users or events are created, data will also be added into the remote database MySQL. And when requesting them, it needs to make interaction with the remote one. As to chatting, the APP will interact with the local database SQLite.

11.Exception Handling

We consider to add Exception Handling mechanism into our project. We name it CustomException, which can print exception number and message, and fix exception in a reasonable way.
Besides, We use two enumeration class: ExceptionNumEnum, which is used for exception numbers, and ExceptionMsgEnum, which is for exception message.

When an exception appears, the application should not exit directly, while it should shows a dialog to alert the user that he/she has done something illegal. The dialog should also shows instruction to tell the user what to do next.